Facultat Informàtica de Barcelona (FIB) – Universitat Politècnica de Catalunya (UPC)

Comunidad virtual y aplicación móvil de seguimiento de precios

Trabajo Final de Grado – Grado de Ingeniería Informática especialidad Ingeniería del software.

Alumno: Marcos López Gallego
Director: Javier Bejar Alonso

27/06/2014

Este proyecto pretende construir un sistema básico con el que poder construir una comunidad virtual de seguimiento de precios. Para tal cometido se analizara, diseñara e implementara la infraestructura básica y la implementación de dos prototipos como ejemplo, un prototipo web y otro Android. El sistema es capaz de registrar un usuario, añadir productos y asociarles establecimientos definiendo su precio y también podrán realizar la actualización de dicho precio.

Uno de los motivos por los que se crea este proyecto es debido al actual contexto socio-económico donde se ha visto reducido la capacidad de ahorro y aumentadas las dificultades económicos por culpa de la actual crisis económica. En este proyecto, intentamos aportar nuestro granito de arena para poder facilitar una poco más la vida de las personas. Proporcionando un sistema en el que se pueda construir una comunidad de seguimiento de precios. De esta manera poder facilitar la tarea de ir a la busca y captura de los productos más económicos en función de la localización geográfica.

Debido al auge de los dispositivos móviles, smartphones, se proporciona un servicio distribuido para poder construir aplicaciones móviles, ya sea para Android, IOS, Windows Mobile y otros sistemas operativos que existen en la actualidad para smartphones. Por esta razón uno de los prototipos será una aplicación Android.

Durante el proyecto se han analizado los posibles riesgos y soluciones, el impacto social y su coste.

Finalmente he dejado constancia de los posibles trabajos futuros a realizar que se podrían hacer a partir de este proyecto.

This project pretend to build a basic system with which to build a virtual community of price monitoring. For this task I analyze, design and implement basic infrastructure and implementation of two prototypes as an example, a web prototype and other Android. The system is able to register a user, add products and associate various properties defining its price and that price updating by users.

One of the reasons why this project is created due to the current socio-economic context that has been reduced savings and increased capacity the economic difficulties because of the current economic crisis. In this project, we intend to contribute our bit. Providing a system that can construct a prices tracking community. Thus to facilitate the task of going to hunt for the cheapest products based on geographic location.

With the rise of mobile devices, Smartphone is provided in order to build a distributed application service, whether for Android, IOS, Windows Mobile and other operating systems that exist today for Smartphone. For this reason one of the prototypes will be an Android application.

During the project we have analyzed the potential risks and solutions, social impact and cost.

Finally, I left evidence of possible future work that could be made from this project.

Agradecimientos

Quiero agradecer a todas aquellas personas que han colaborado directa o indirectamente en este proyecto.

En especial a mi director del Trabajo Final de Grado que me ha ofrecido la oportunidad de realizar este proyecto.

Para terminar, agradecer a todo el profesorado de la FIB por realizar su labor docente y permitir que alumnos como yo podamos salir del grado con valiosos conocimientos que nos ayudaran en el día a día de nuestra carrera profesional.

Índice

1. Introducción 12
   1.1. Problema 12
2. Objetivos 12
   2.1. Objetivo Principal 12
   2.2. Objetivo secundario 12
3. Partes interesadas (Stakeholders) 13
   3.1. El promotor 13
   3.2. El cliente 13
   3.3. Usuario 13
   3.4. Otros Stakeholders 13
      3.4.1. Proveedores 13
      3.4.2. Diseñadores del sistema 14
      3.4.3. Diseñadores gráficos 14
      3.4.4. Sociedad 14
   3.5. Quien lo utilizara 14
   3.6. Como se beneficiaran 15
4. Estado del Arte 15
   4.1. Estado de la población española 15
      4.1.1. Ingresos medios (1) 15
      4.1.2. Población en riesgo de pobreza (1) 15
      4.1.3. Situación económica de los hogares (1) 16
   4.2. Casos similares 17
      4.2.1. El comparador(2) 17
      4.2.2. Carritus (4) 17
   4.3. Conclusiones 18
5. Alcance 18
6. Posibles obstáculos 19
7. Metodología y rigor 20
   7.1. Metodología a utilizar durante el proyecto 20
   7.2. Método del proyecto 20
   7.3. Herramientas de seguimiento 20
   7.4. Método de validación 20
8. Contexto 21
   8.1. Eventos de negocio principales 21

8.2. Diagrama de contexto 22
8.3. Diagramas de actividad 23
8.3.1. Buscar un producto 23
8.3.2. Añadir precio 24
8.3.3. Añadir establecimiento 25
9. Glosario 26
10. Especificación de requisitos 26
10.1. Requisitos funcionales 26
10.1.1. Diagrama casos de uso 26
10.1.2. Descripción de los casos de uso 28
10.1.3. Diagrama conceptual 34
10.2. Requisitos de calidad (no funcionales) 34
10.2.1. Requisitos de estilo y apariencia 34
10.2.2. Requisitos de usabilidad y humanidad 35
10.2.3. Requisitos de rendimiento 37
10.2.4. Requisitos de seguridad 39
10.2.5. Requisitos operativos 39
10.2.6. Requisitos de actuación 40
11. Arquitectura del sistema (7) 40
11.1. Arquitectura N-Capas con Orientación al dominio 41
11.1.1. Descripción de las capas 42
11.2. Patrones utilizados 43
11.3. Capa de dominio 44
11.3.1. Diagrama de clases 44
11.3.2. Contratos de los repositorios 44
11.3.3. Capa de servicios 46
11.4. Capa de infraestructura de datos (DAL) 49
11.4.1. Modelo de datos 49
11.4.2. Implementación de repositorios 50
11.5. Capa de aplicación 51
11.5.1. Capa de servicios 51
11.5.2. Capa de DTOs 54
11.5.3. Capa de adaptadores 54
11.5.4. Capa de agentes 55
11.6. Capa de infraestructura 56
11.6.1. Capa de comunicaciones 56
11.6.2. Capa de mapas geográficos 58

11.6.3. Capa de adaptadores 60
11.6.4. Capa de seguridad 61
11.6.5. Capa de utilidades 64
11.7. Capa de servicios distribuidos 65
11.7.1. Capa de controladores 65
11.8. Capa de presentación 68
11.8.1. Aplicación web 68
11.8.2. Cliente Android 78
12. Planificación 83
12.1. Planificación inicial 85
12.2. Planificación revisada 86
12.3. Valoración de alternativas y plan de acción 87
13. Presupuesto 88
13.1. Consideraciones 88
13.2. Control presupuestario 88
13.3. Presupuesto de recursos humanos 88
13.4. Presupuesto del hardware 88
13.5. Presupuesto del software 89
13.6. Presupuesto licencias 89
13.7. Presupuesto de los proveedores 89
13.8. Costes imprevistos 89
13.9. Presupuesto total y viabilidad 90
13.10. Impacto social y medioambiental 90
13.10.1. Impacto social 90
13.10.2. Impacto medioambiental 90
14. Conclusiones y trabajo futuro 91
14.1. Conclusiones 91
14.2. Trabajo futuro 92
15. Bibliografía 93

Índice de Ilustraciones

Ilustración 1: Diagrama de contexto. 22
Ilustración 2: Diagrama de actividad de la búsqueda de un producto. 23
Ilustración 3: Diagrama de actividad para añadir un precio. 24
Ilustración 4: Diagrama de actividad para añadir un establecimiento. 25
Ilustración 5: Casos de uso. 27
Ilustración 6: Diagrama conceptual. 34
Ilustración 7: Arquitectura N-capas con orientación al dominio. 41
Ilustración 8: Capa del dominio - Diagrama de clases. 44
Ilustración 9: Capa del dominio - Diagrama de contratos de los repositorios. 44
Ilustración 10: Capa del dominio - Diagrama de servicios. 46
Ilustración 11: Diagrama del modelo de datos. 49
Ilustración 12: Diagrama de la subcapa de servicios de la capa aplicación. 51
Ilustración 13: Diagrama de DTOs. 54
Ilustración 14: Traductor de direcciones a coordenadas geográficas. 55
Ilustración 15: capa de comunicaciones de la infraestructura. 56
Ilustración 16: Diagrama de la capa de mapas geográficos. 58
Ilustración 17: Diagrama de la capa de adaptadores capa de infraestructura. 60
Ilustración 18: Diagrama capa de seguridad. 61
Ilustración 19: Diagrama de capas de las utilidades de la capa infraestructura. 64
Ilustración 20: Diagrama capa de controladores. 65
Ilustración 21: Diagrama de representación del modelo de datos en la aplicación web. 69
Ilustración 22: Diagrama de controladores de la aplicación web. 69
Ilustración 23: Vista LogIn aplicación web. 72
Ilustración 24: Vista Registro de UsuariosController de la aplicación web. 73
Ilustración 25: Vista Index de UsuariosController de la aplicación web. 74
Ilustración 26: Vista EditarDireccion de UsuariosController de la aplicación web. 74
Ilustración 27: Vista InsertarDireccion de UsuariosController de la aplicación web. 75
Ilustración 28: Vista Index de ProductosController de la aplicación web. 75
Ilustración 29: Vista Index de EstablecimientosController de la aplicación web. 76
Ilustración 30: Vista Ver de EstablecimientosController de la aplicación web mostrando resultados. 76
Ilustración 31: Vista Ver de EstablecimientosController de la aplicación web para añadir producto o actualizar precio. 77
Ilustración 32: Vista Insertar de EstablecimientosController de la aplicación web. 77
Ilustración 33: Diagrama de la capa de representación del modelo en Android. 78
Ilustración 34: Diagrama de agentes del cliente Android. 78
Ilustración 35: Diagrama de controladores del cliente Android. 80
Ilustración 36: Vista login del cliente Android. 81
Ilustración 37: Vista tab productos del cliente Android. 81
Ilustración 38: Vista tab establecimiento. 82
Ilustración 39: Vista tab del usuario del cliente Android. 82
Ilustración 40: Vista Registro del cliente Android. 83
Ilustración 41: Planificación Inicial. 85
Ilustración 42: Planificación revisada. 86

Índice de Gráficos

Gráfico 1: Evolución de los ingresos medios por hogar. 15
Gráfico 2: Tasa de riesgo de pobreza por edad..... 16
Gráfico 3: Evolución de las dificultades económicas de los hogares. 17

Índice de Tablas

Tabla 1: Posibles obstáculos. 19
Tabla 2: Eventos de negocio principales. 22
Tabla 3 : Caso de uso para el registro. 28
Tabla 4: Caso de uso de login (Autenticación). 28
Tabla 5: Caso de uso para modificar los datos del usuario. 29
Tabla 6: Caso de uso para buscar los precios de un producto en los establecimientos cercanos. 29
Tabla 7: Caso de uso para buscar establecimientos. 30
Tabla 8: Caso de uso para añadir una dirección del usuario. 30
Tabla 9: Caso de uso para editar una dirección del usuario. 30
Tabla 10: Caso de uso para borrar una dirección del usuario. 31
Tabla 11: Caso de uso para ver los productos de un establecimiento. 31
Tabla 12: Añadir un establecimiento al producto. 32
Tabla 13: Caso de uso para añadir un producto a un establecimiento. 32
Tabla 14: Caso de uso para añadir un producto al sistema. 32
Tabla 15: Caso de uso para añadir un precio al producto. 33
Tabla 16: Caso de uso para añadir un nuevo establecimiento al sistema. 33
Tabla 17: Requisito estilo y apariencia. 35
Tabla 18: Requisito de facilidad de uso. 35
Tabla 19: Requisito del idioma. 36
Tabla 20: Requisito de los iconos. 36
Tabla 21: Requisito de accesibilidad. 37
Tabla 22: Requisito de velocidad y latencia. 37
Tabla 23: Requisito de la capacidad en número de usuarios. 38
Tabla 24: Requisito de la disponibilidad. 38
Tabla 25: Requisito de escalabilidad. 39
Tabla 26: Requisito de seguridad. 39
Tabla 27: Requisito de actualizaciones sin errores. 40
Tabla 28: Requisito de privacidad. 40
Tabla 29: Interfaz IBaseRepository<T>. 45
Tabla 30: Interfaz IProductoRepository. 45
Tabla 31: Interfaz IEstablecimientoRepository. 45
Tabla 32: Interfaz IUsuarioRepository. 46
Tabla 33: Interfaz IProductoService. 47
Tabla 34: Interfaz IDireccionService. 47
Tabla 35: Interfaz IUsuarioService. 47
Tabla 36: Interfaz IEstablecimientoService. 48
Tabla 37: clase ProductoService. 48
Tabla 38: clase DireccionService. 48
Tabla 39: Clase UsuarioService. 48
Tabla 40: Clase EstablecimientoService. 48
Tabla 41: Diagrama de implementación de los repositorios. 50
Tabla 42: Clase BaseRepository. 50
Tabla 43: Clase ProductoRepository. 50
Tabla 44: Clase EstablecimientoRepository. 50
Tabla 45: Clase UsuarioRepository. 51

Tabla 46: Interfaz IEstablecimientoService capa de aplicación. 52
Tabla 47: Interfaz IUsuarioService capa de aplicación. 52
Tabla 48: Interfaz IProductoService capa de aplicación. 53
Tabla 49: Clase EstablecimientoService capa de aplicación. 53
Tabla 50: Clase ProductoService capa de aplicación. 53
Tabla 51: Clase UsuarioService capa de aplicación. 53
Tabla 52: Interfaz ITraductorDireccion. 55
Tabla 53: Clase TraductorDireccion. 55
Tabla 54: Interfaz IWebRequest. 56
Tabla 55: Clase WebRequestWrapper. 57
Tabla 56: Clase estática GoogleMaps. 59
Tabla 57: Clase GeocodingEngine. 59
Tabla 58: Interfaz IMappingService. 60
Tabla 59: Clase MappingService. 60
Tabla 60: Clase estática Encode. 62
Tabla 61: Clase TokenEncodeInspector. 63
Tabla 62: Clase TokenCertificateInspector. 63
Tabla 63: Clase HttpsGuard. 63
Tabla 64: Clase CryptographyHelper. 64
Tabla 65: Clase GeoCalculator. 65
Tabla 66: Clase controladora AuthorizeController. 66
Tabla 67: Clase controladora UsuarioController. 66
Tabla 68: Clase controladora ProductosController. 66
Tabla 69: Clase controladora PreciosController. 67
Tabla 70: Clase controladora DireccionUsuarioController. 67
Tabla 71: Clase controladora EstablecimientoController. 68
Tabla 72: Clase controladora de la web UsuariosController. 71
Tabla 73: Clase controladora de la web ProductosController. 71
Tabla 74: Clase controladora de la web EstablecimientosController. 72
Tabla 75: Clase WebService del cliente Android. 79
Tabla 76: Clase LoginActivity del cliente Android. 80
Tabla 77: Clase TabsActivity del cliente Android. 80
Tabla 78: Clase RegistroActivity del cliente Android. 80
Tabla 79: Presupuesto de recursos humanos. 88
Tabla 80: Presupuesto del hardware. 88
Tabla 81: Presupuesto del software. 89
Tabla 82: Presupuesto licencias. 89
Tabla 83: Presupuesto de los proveedores. 89

1. Introducción

1.1. Problema

En la actualidad, donde el contexto socio-económico castiga al ciudadano de clase media-baja debido a los recortes presupuestarios del gobierno, la alta tasa de desempleo y la crisis económica. Cada vez es más difícil poder llegar a final de mes y cubrir las necesidades básicas como la alimentación, energía, etc. La situación se agrava con la constante subida de impuestos tanto directos como indirectos reduciendo el poder adquisitivo de las familias. De esta situación surge la necesidad de tener que buscar los bienes básicos con el precio más reducido posible.

En la actualidad para poder encontrar el precio más reducido hay que realizar una dura tarea, que puede llegar a consumir mucho tiempo al tener que entrar en los establecimientos y anotarse el precio de los productos buscando el más económico.

Existen productos que se compran con frecuencia y cuyo precio puede variar ampliamente de un establecimiento comercial a otro. El seguimiento de los precios de los productos que se compran con frecuencia resulta una tarea complicada de hacer en solitario.

2. Objetivos

2.1. Objetivo Principal

El objetivo principal del proyecto es definir, diseñar e implementar un sistema que sirva como base que permita construir una comunidad virtual de usuarios para poder seguir e informar de los precios de los establecimientos comerciales para que los usuarios se puedan ayudar mutuamente.

El proyecto definirá una estructura base para poder realizar una comunidad virtual de seguimiento de precios. Con esta estructura se podrá crear una comunidad añadiendo los requisitos que se deseen a posteriori para completar y conseguir los objetivos que se definan para un sistema de seguimiento completo.

El objetivo final seria utilizar esta infraestructura desarrollada para implementar los dos prototipos, la web y la aplicación Android.

2.2. Objetivo secundario

Reducir el ratio de horas invertidas en el seguimiento de precios con el volumen de información disponible (Tiempo consumido/Número de productos comparados) que debe invertir el usuario.

Este objetivo sólo se puede cumplir gracias a la colaboración entre usuarios al compartir la información dentro del sistema.

3. Partes interesadas (Stakeholders)

3.1. El promotor

El principal promotor es el creador del propio sistema y el director del proyecto. Son los máximos responsables del sistema teniendo la capacidad de decidir sobre la tecnología, los servicios que ofrecerá, el nivel de calidad y búsqueda de recursos consiguiendo un sistema que requiera un coste mínimo con un mantenimiento mínimo.

3.2. El cliente

En este caso es el mismo promotor.

3.3. Usuario

El usuario es el principal beneficiario del sistema. Utilizara el sistema para invertir menos tiempo en el seguimiento de precios y poder obtener los productos al precio más económico posible.

3.4. Otros Stakeholders

3.4.1. Proveedores

Debemos diferenciar los dos tipos de proveedores necesarios.

Proveedor de hospedaje web

Es el que nos ofrecerá las infraestructuras necesarias para implantar el sistema en internet ofreciéndonos un alojamiento web y un dominio donde desplegar nuestro servicio. La información que se debe de tener en cuenta es la siguiente.

- Tipos y características de los servidores disponibles.
- Lenguajes posibles a utilizar.
- Nombre del dominio.
- Tipo de base de datos.

Proveedor de posición geográfico

Es quien proporcionara una forma de poder traducir direcciones postales en puntos geográficos para poder geoposicionar y realizar cálculos de proximidad.

3.4.2. Diseñadores del sistema

Quienes diseñaran el sistema, serán ingenieros del software que tienen que conocer la especificación de requisitos para poder diseñar el sistema correctamente.

3.4.3. Diseñadores gráficos

Son los encargados de diseñar gráficamente las interfaces gráficas. Para este cometido necesitan saber la siguiente información.

- Numero de pantallas.
- Tecnologías y dispositivos donde estarán las interfaces graficas.
- Pantallas y sus funcionalidades.

3.4.4. Sociedad

El sistema de seguimiento tendrá un impacto económico en la sociedad favoreciendo el ahorro de los hogares con dificultades económicas.

3.5. Quien lo utilizara

La utilización de la comunidad virtual está orientada a usuarios de una clase social media y baja. Son usuarios que debido al contexto económico tienen dificultades para llegar a final de mes.

Los únicos requisitos para poder utilizar el producto son los descritos a continuación.

- Acceso a una conexión a internet
- Conocimientos básicos de la utilización de internet.
- Un ordenador o dispositivo móvil a su alcance.

Los usuarios serán personas de perfiles diferentes, desde jóvenes solteros hasta padres y madres.

3.6. Como se beneficiaran

Los usuarios que pertenezcan a la comunidad virtual se podrán beneficiar gracias a la aportación del resto de usuarios, minimizando el tiempo de búsqueda y comparación de precios.

Los usuarios gracias a sus pequeñas aportaciones ayudaran al conjunto de la comunidad.

4. Estado del Arte

4.1. Estado de la población española

4.1.1. Ingresos medios (1)

Los resultados provisionales de la Encuesta de Condiciones de Vida (ECV) (1) del año 2013 ofrecen información sobre los ingresos medios de los hogares durante el año 2012. Según estos resultados, el ingreso monetario medio anual neto por hogar se situó en 23.123 euros, con una disminución del 3,5% respecto al año anterior.

El ingreso medio por persona alcanzó los 9.098 euros, cifra un 2,4% inferior a la registrada el año precedente.

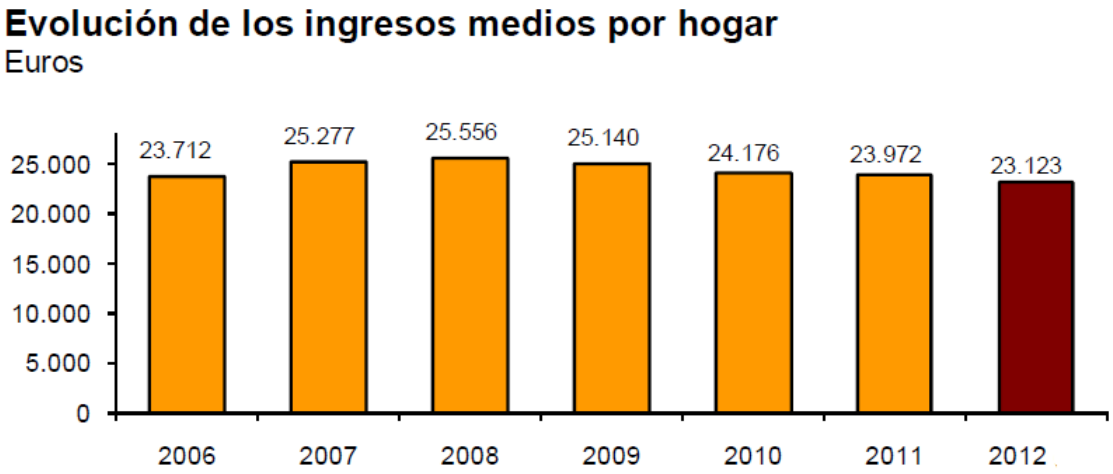


Gráfico 1: Evolución de los ingresos medios por hogar.

4.1.2. Población en riesgo de pobreza (1)

Aunque los ingresos medios disminuyen, el porcentaje de población por debajo del umbral de pobreza también se reduce respecto al año anterior. La población en riesgo de pobreza es un indicador relativo que mide desigualdad. No mide pobreza absoluta sino cuántas personas tienen ingresos bajos en relación al conjunto de la población.

Así, en 2013 la tasa de riesgo de pobreza se sitúa en el 21,6% de la población residente en España, frente al 22,2% registrado el año anterior.

Cabe destacar la disminución de esta tasa entre los mayores de 65 años (de 2,6 puntos entre 2012 y 2013). Por su parte, la tasa de los menores de 16 años baja 1,2 puntos.

Tasa de riesgo de pobreza por edad

Porcentajes

	2009	2010	2011	2012	2013 (1)
TOTAL	20,1	21,4	22,2	22,2	21,6
Menos de 16 años	26,5	28,3	28,7	28,9	27,7
De 16 a 64 años	17,9	20,1	21,3	22,4	22,5
65 y más años	23,1	20,5	19,5	14,8	12,2

Gráfico 2: Tasa de riesgo de pobreza por edad.

4.1.3. Situación económica de los hogares (1)

El 16,9% de los hogares españoles manifiesta llegar a fin de mes con “mucha dificultad” en 2013. Este porcentaje supera en 3,4 puntos al registrado el año anterior.

Por su parte, el 40,9% de los hogares no tiene capacidad para afrontar gastos imprevistos, frente al 41,4% del año 2012.

El 45,8% de los hogares no se puede permitir ir de vacaciones fuera de casa al menos una semana al año. Este porcentaje es 0,7 puntos mayor que el registrado en 2012.

El 9,2% de los hogares tiene retrasos en los pagos a la hora de abonar gastos relacionados con la vivienda principal (hipoteca o alquiler, recibos de gas, electricidad, comunidad,...) en los 12 meses anteriores al de la entrevista. Este porcentaje es 0,8 puntos mayor que el del año anterior.

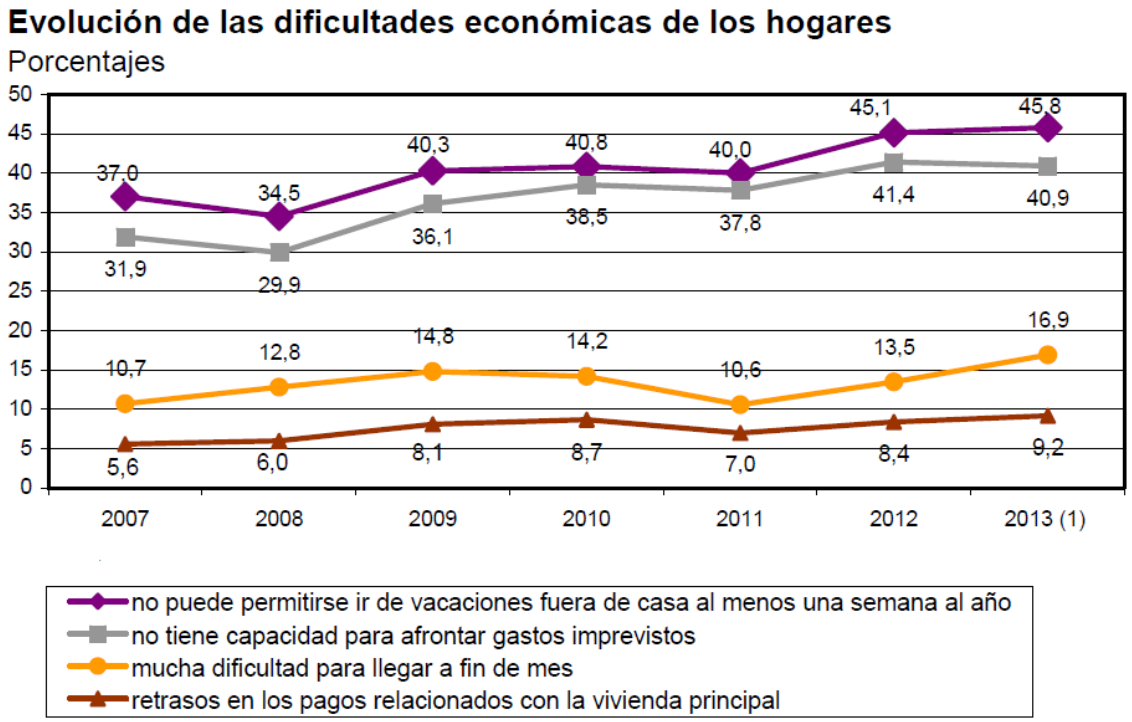


Gráfico 3: Evolución de las dificultades económicas de los hogares.

4.2. Casos similares

4.2.1. El comparador(2)

"El Comparador" es un comparador online propiedad de la cadena de supermercados Caprabo.

Este comparador se limita a comparar los precios de la cadena de supermercados Caprabo con la de Mercadona. Es claramente una estrategia de marketing (3) para la fidelización de clientes en Caprabo.

4.2.2. Carritus (4)

Es un comparador de precios de los supermercados siguientes.

- Mercadona
- Caprabo
- Eroski
- Condis
- El corte Ingles
- Carrefour
- Hipercor
- Alcampo

Carritus realiza la comparación de precios en función de una lista de la compra y no de los productos de los supermercados anteriormente citados, permitiendo realizar la compra online desde su plataforma web teniendo como prerrequisito el registro en la web del supermercado final a comprar.

4.3. Conclusiones

Hay múltiples comparadores de precios que se basan en los precios mostrados vía web de los distintos establecimientos. Establecimientos que por norma general son grandes superficies descartando a pequeños comercios.

Algunos son usados como herramientas de marketing y algunos otros como intermediaros pero ninguno se basa en compartir esfuerzos de distintos usuarios formando una comunidad virtual para ayudarse mutuamente donde se tendrían en cuenta todos los establecimientos introducidos por las aportaciones de los usuarios pertenecientes a la comunidad.

5. Alcance

El alcance de este proyecto es definir, diseñar e implementar un sistema base que permita construir una comunidad de usuarios para poder seguir e informar de los precios de los establecimientos comerciales para que los usuarios se puedan ayudar mutuamente.

Para terminar, se utilizara esta infraestructura desarrollada para implementar dos prototipos.

6. Posibles obstáculos

A continuación se enumeran los posibles obstáculos que podrían surgir durante el proyecto y que se han de tener en cuenta.

Riesgo	Peligrosidad	Medidas de prevención	Soluciones
Elegir una arquitectura o un diseño erróneo que pueda dificultar el uso del software.	Alta	Detallar lo máximo posible las funcionales y casos de uso.	Si no es posible solucionar el problema que causa un diseño erróneo buscar la alternativa que se aproxime más a la solución.
El software podría no adecuarse correctamente a las necesidades de los usuarios.	Alta	Hablar con el mayor número posible de usuarios potenciales.	Revisar y redefinir los requisitos que no se adecuen correctamente.
El proyecto podría ser demasiado amplio para el tiempo disponible.	Media	Identificar y definir primero aquellas funcionalidades indispensables.	Establecer un punto de corte de funcionales teniendo en cuenta su importancia.
Dificultades para poder encontrar las soluciones adecuadas durante el periodo de diseño e implementación.	Baja	Documentarse adecuadamente sobre Patrones de diseño y formas de implementarlas.	Buscar la unificación y personalización de aquellos patrones que sean más adecuados.
Perdida de archivos.	Media	Realizar 2 copias diarias de toda la información. Cada copia debe estar en dispositivos de almacenamiento distintos para tener siempre disponible una versión.	Recuperar la última copia.

Tabla 1: Posibles obstáculos.

7. Metodología y rigor

7.1. Metodología a utilizar durante el proyecto

Se utilizara la metodología Scrum (5) adaptada al proyecto. Al ser un único integrante se realizaran Sprints de 2 semanas pero sin la reuniones diarias al no tener sentido en este caso.

7.2. Método del proyecto

Debido a la naturaleza del proyecto se deben utilizar diversos entornos y lenguajes de programación.

Para crear el sistema base de la comunidad se utilizara una arquitectura de tipo SaaS para facilitar la multiplataforma, de este manera solo se necesitara un cliente por cada plataforma. Para desarrollar la parte del servidor se utilizara tecnología .NET por el desarrollo rápido que proporciona y para ampliar los conocimientos en dicha tecnología. La plataforma que se elige para crear la aplicación móvil es Android, se utilizara el lenguaje java.

7.3. Herramientas de seguimiento

Para la planificación del proyecto se utilizaran herramientas como gantter, que permite realizar diagramas temporales con los objetivos parciales. También se utilizara Team Foundation Server (6) para la planificación y control del código fuente en .NET. Para tener un control del código fuente de la aplicación Android se utilizara Git.

Se llevarán a cabo reuniones con el director del proyecto, en las cuales se discutirá el estado del proyecto para ver si se está avanzando en la dirección correcta y si se están cumpliendo con los plazos establecidos.

Se hará uso de servicios como el correo electrónico para resolver dudas puntuales que puedan surgir durante la realización del trabajo.

7.4. Método de validación

A continuación se describen los procesos de validación de las diferentes fases del desarrollo del software.

- **Ingeniería de Requisitos.**
  - Una vez finalizado el documento de ingeniería de requisitos contrastarlo con el director del proyecto y los Stakeholders.

- **Análisis conceptual.**
  - Comprobar con el documento de ingeniería de requisitos que se cumplen todos los requisitos.
- **Diseño del software.**
  - Comprobar con el documento de ingeniería de requisitos y análisis conceptual que se cumplen todos los requisitos.
- **Implementación.**
  - Realizar test unitarios en un mínimo del 50% del software.
  - Realizar test de integración en un mínimo del 80% del software.

8. Contexto

8.1. Eventos de negocio principales

Nombre del evento	Entradas y Salidas	Resumen
1. Añadir precio.	**Entrada:** Se indica un producto, establecimiento y precio.	El sistema guarda la información del nuevo precio asociado a un establecimiento y producto incluyendo su fecha.
2. Buscar producto.	**Entrada:** Se indica un producto y una dirección.	Se buscan los establecimientos más cercanos el producto y se muestra el último precio añadido.
3. Ver productos de un establecimiento.	**Entrada:** Se indica un establecimiento a ver.	Se buscan los establecimientos más cercanos, se selecciona el deseado y muestra los productos con el último precio añadido.
4. Añadir establecimiento al producto.	**Entrada:** Se indica nuevo establecimiento en el producto.	Se añade un nuevo establecimiento a un producto indicando su precio.
5. Añadir producto.	**Entrada:** Se indica establecimiento, producto y su precio.	Se añade un nuevo producto a un establecimiento indicando el precio.

Nombre del evento	Entradas y Salidas	Resumen
6. Buscar establecimiento.	**Entrada:** Se indica una dirección.	Se buscan los establecimientos más cercanos a la dirección.
7. Traducir dirección.	**Salida:** Se introduce una dirección.	Se traduce a coordenadas geográficas una dirección.

Tabla 2: Eventos de negocio principales.

8.2. Diagrama de contexto

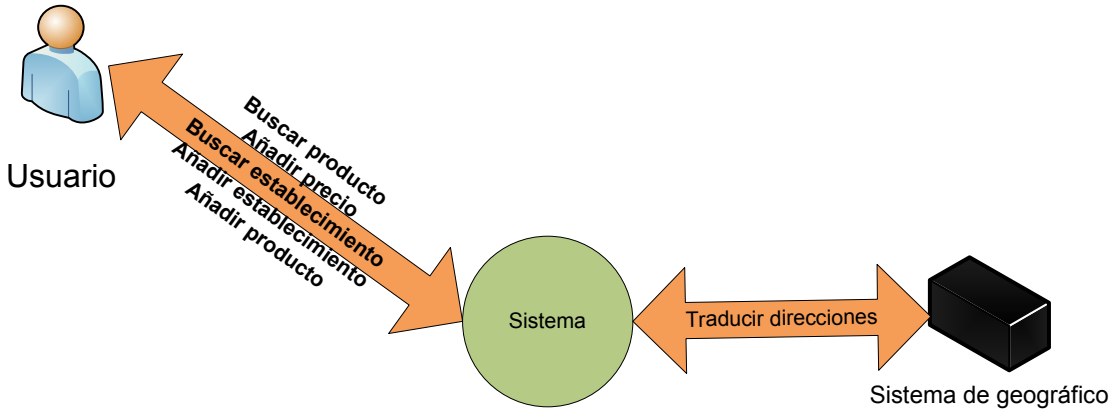


Ilustración 1: Diagrama de contexto.

8.3. Diagramas de actividad

8.3.1. Buscar un producto

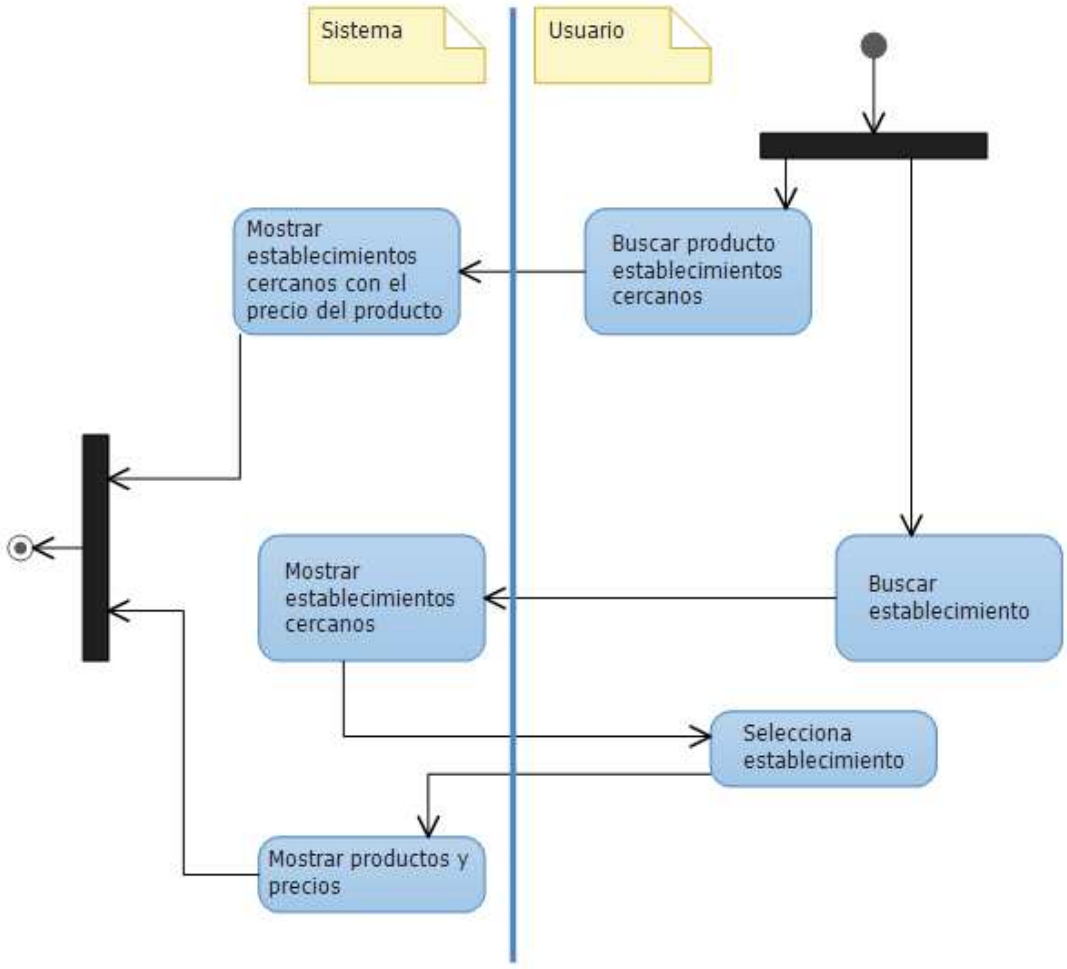


Ilustración 2: Diagrama de actividad de la búsqueda de un producto.

8.3.2. Añadir precio

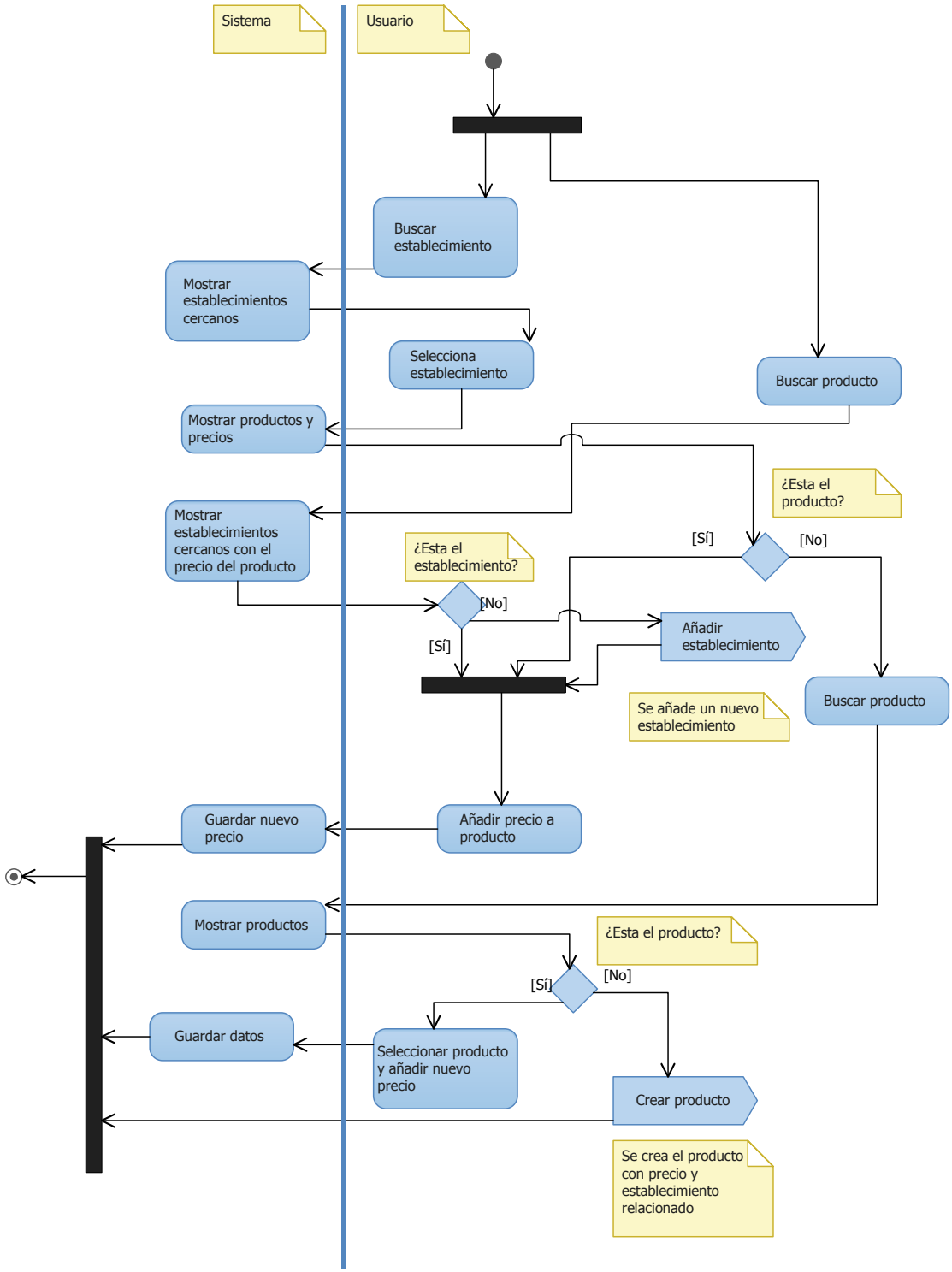


Ilustración 3: Diagrama de actividad para añadir un precio.

8.3.3. Añadir establecimiento

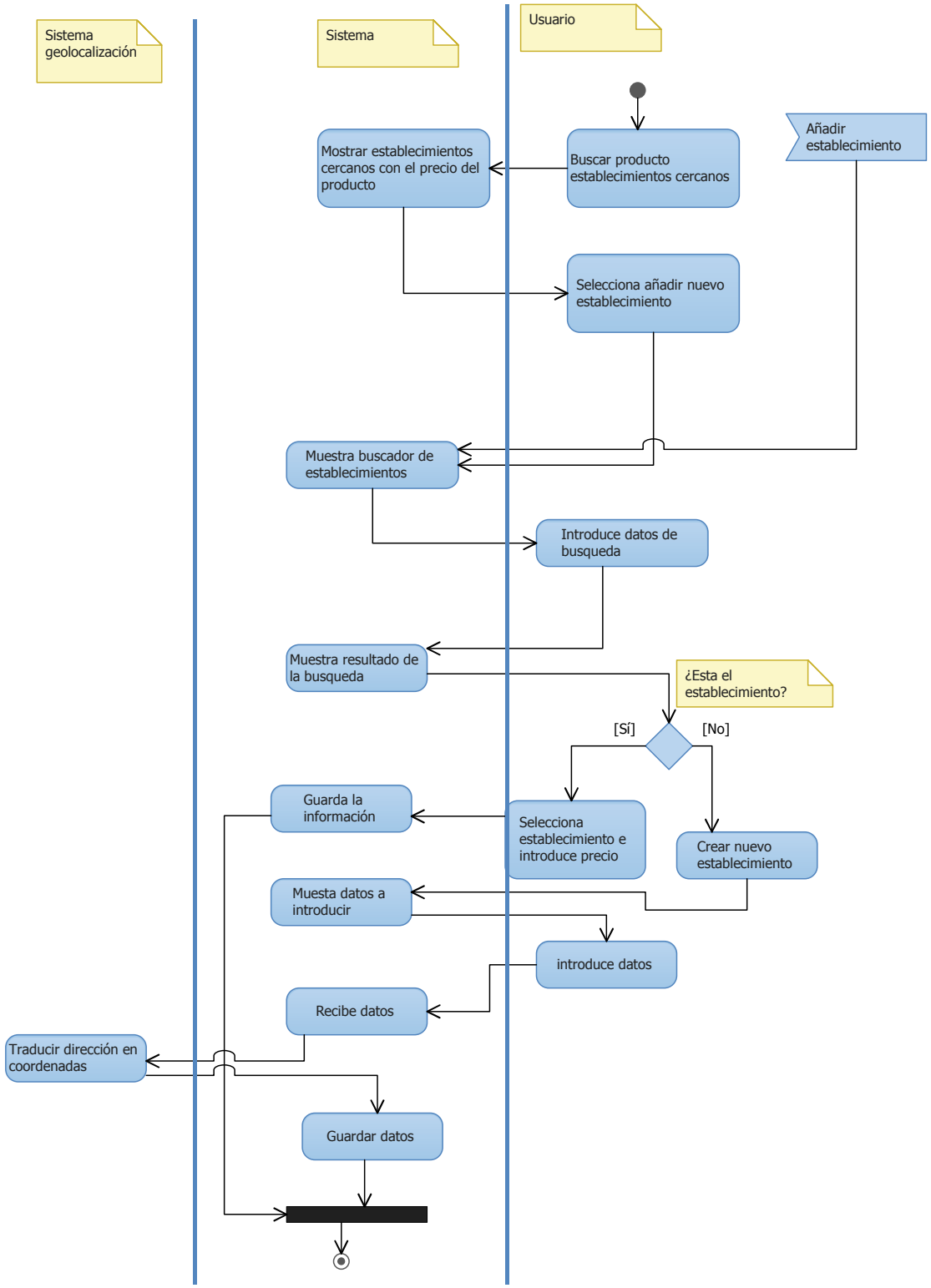


Ilustración 4: Diagrama de actividad para añadir un establecimiento.

9. Glosario

En el glosario se definen aquellas palabras o signos que no queden claramente bien definidas evitando ambigüedades o que el lector necesita saber para poder entender su significado de una forma clara.

- **Comunidad virtual:** Lugar alojado en internet donde los usuarios forman una comunidad para ayudarse mutuamente con un objetivo común.
- **DTO:** Objeto de transferencia de datos. Es un tipo de objeto que es utilizado para transportar datos entre procesos.
- **Establecimiento:** Lugar físico donde se ofrecen productos a cambio de un intercambio monetario.
- **Login:** Autentificar a una usuario o sistema.
- **Nickname:** Alias.
- **Producto:** Es todo aquel bien de consumo.
- **Sprint:** Iteración, periodo de tiempo.
- **Stakeholders:** grupos o entornos que puedan estar interesados y puedan afectar o estar afectados por el proyecto.
- **Token:** Un token es una cadena de caracteres con un significado coherente.
- **[→Texto]:** Indica que se ejecuta otro caso de uso, en este caso llamado “texto”.

10. Especificación de requisitos

10.1. Requisitos funcionales

10.1.1. Diagrama casos de uso

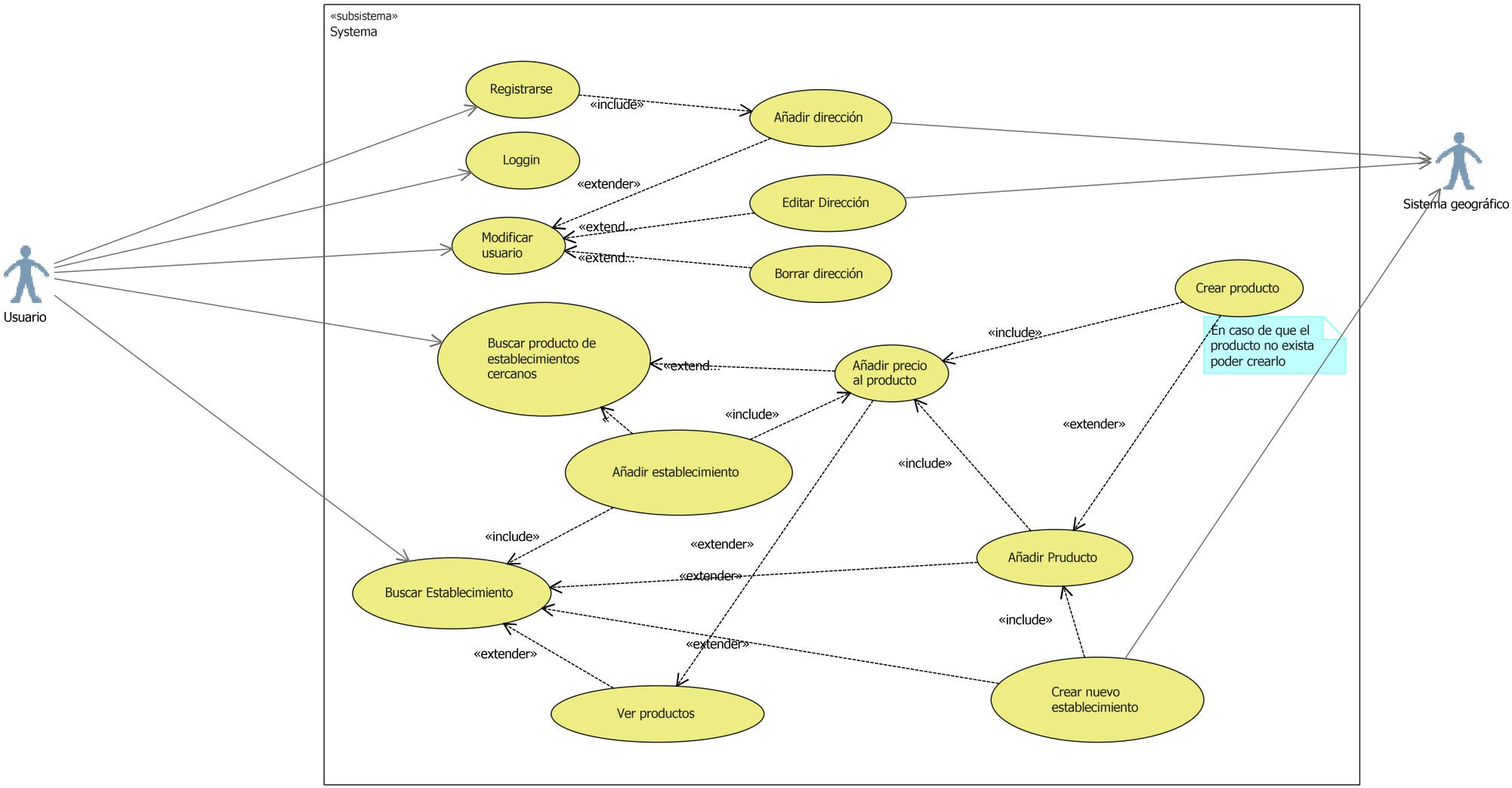


Ilustración 5: Casos de uso.

10.1.2. Descripción de los casos de uso

Caso de uso	Registrarse.
Actor principal	Usuario.
Precondición	El actor no está identificado en el sistema.
Disparador	Un usuario quiere registrarse en el sistema.
Escenario principal	
1. El sistema pide al actor un alias (nickname), contraseña, un correo electrónico y una dirección a añadir. [→Añadir dirección] 2. El actor introduce los datos solicitados. 3. El sistema comprueba los datos introducidos, los almacena y autentifica en el sistema. [→Login] 4. El caso de uso termina.	
Extensiones	
4a. Alguno de los datos introducidos por el actor son incorrectos: 4a1. El sistema notifica al actor que los datos introducidos no son correctos. 4a3. El caso de uso continúa en el paso 1 del escenario principal.	

Tabla 3 : Caso de uso para el registro.

Caso de uso	Login (Autenticación).
Actor principal	Usuario.
Precondición	El actor no está identificado en el sistema.
Disparador	Un usuario desea entrar en el sistema.
Escenario principal	
1. El sistema pide al actor sus credenciales. 2. El actor introduce los datos solicitados. 3. El sistema comprueba los datos introducidos. 4. El sistema le muestra la pantalla de inicio. 5. El caso de uso termina.	
Extensiones	
3a. Alguno de los datos introducidos por el actor son incorrectos: 3a1. El sistema notifica al actor que los datos introducidos no son correctos. 3a3. El caso de uso continúa en el paso 1 del escenario principal.	

Tabla 4: Caso de uso de login (Autenticación).

Caso de uso	Modificar datos del usuario.
Actor principal	Usuario.
Precondición	El actor está identificado en el sistema.
Disparador	Un usuario desea modificar sus datos.
Escenario principal	

1. El sistema muestra la información almacenada. 2. El actor modifica los datos que desea. 3. El sistema comprueba los datos introducidos. 4. El sistema guarda la información modificada. 5. El sistema le muestra la pantalla de inicio. 6. El caso de uso termina.
Extensiones
2b. El actor desea añadir una nueva dirección. [→Añadir dirección] 2b. El actor desea editar una dirección. [→Editar dirección] 2b. El actor desea borrar una dirección. [→Borrar dirección] 3a. Alguno de los datos introducidos por el actor son incorrectos: 3a1. El sistema notifica al actor que los datos introducidos no son correctos. 3a3. El caso de uso continúa en el paso 1 del escenario principal.

Tabla 5: Caso de uso para modificar los datos del usuario.

Caso de uso	Buscar un producto en los establecimientos cercanos.
Actor principal	Usuario.
Precondición	El actor está identificado en el sistema.
Disparador	Un usuario desea encontrar el precio de un producto de los establecimientos más cercanos.
Escenario principal	
1. El actor selecciona una dirección, un producto y la distancia máxima. 2. El sistema muestra los establecimientos por orden de precio de más bajo al más alto. 3. El caso de uso termina.	
Extensiones	
3a. El actor desea añadir un nuevo precio al producto [→Añadir precio al producto]. 3b. El actor desea añadir un nuevo establecimiento al producto [→Añadir establecimiento].	

Tabla 6: Caso de uso para buscar los precios de un producto en los establecimientos cercanos.

Caso de uso	Buscar establecimiento.
Actor principal	Usuario.
Precondición	El actor está identificado en el sistema.
Disparador	Un usuario quiere buscar un establecimiento.
Escenario principal	
1. El actor indica un nombre, dirección y una distancia. 2. El sistema muestra todos los establecimientos encontrados.	

3. El caso de uso termina.
Extensiones
3a. El actor desea añadir un nuevo precio. [→Añadir producto]. 3b. El actor desea añadir un nuevo establecimiento [→Crear nuevo establecimiento]. 3c. El actor desea ver los productos de un establecimiento [→Ver productos].

Tabla 7: Caso de uso para buscar establecimientos.

Caso de uso	Añadir dirección.
Actor principal	Usuario.
Precondición	El actor está identificado en el sistema.
Disparador	Un usuario quiere añadir una nueva dirección.
Escenario principal	
1. El actor indica una dirección. 2. El sistema valida y guarda la dirección. 3. El caso de uso termina.	
Extensiones	
2a. Alguno de los datos introducidos por el actor son incorrectos: 2a1. El sistema notifica al actor que los datos introducidos no son correctos. 2a3. El caso de uso continúa en el paso 1 del escenario principal.	

Tabla 8: Caso de uso para añadir una dirección del usuario.

Caso de uso	Editar dirección.
Actor principal	Usuario.
Precondición	El actor está identificado en el sistema.
Disparador	Un usuario quiere editar una nueva dirección.
Escenario principal	
1. El actor indica una dirección. 2. El sistema valida y guarda la dirección. 3. El caso de uso termina.	
Extensiones	
2a. Alguno de los datos introducidos por el actor son incorrectos: 2a1. El sistema notifica al actor que los datos introducidos no son correctos. 2a3. El caso de uso continúa en el paso 1 del escenario principal.	

Tabla 9: Caso de uso para editar una dirección del usuario.

Caso de uso	Borrar una dirección.

Actor principal	Usuario.
Precondición	El actor está identificado en el sistema.
Disparador	Un usuario quiere borrar una nueva dirección.
Escenario principal	
1. El actor indica una dirección. 2. El sistema valida y elimina la asociación entre el usuario y la dirección. 3. El caso de uso termina.	
Extensiones	
2a. El actor solo tiene una dirección: 3a1. El sistema notifica al actor que es la única dirección que tiene y no puede eliminarla. 3a3. El caso de uso termina.	

Tabla 10: Caso de uso para borrar una dirección del usuario.

Caso de uso	Ver productos.
Actor principal	Usuario.
Precondición	El actor está identificado en el sistema y viene del caso de uso Buscar Establecimiento y haber seleccionado uno
Disparador	Un usuario quiere ver los productos de un establecimiento.
Escenario principal	
1. El actor indica el establecimiento. 2. El sistema muestra todos los productos en orden alfabético con su último precio. 3. El caso de uso termina.	
Extensiones	
3a. El actor quiere añadir un nuevo precio [→Añadir precio al producto].	

Tabla 11: Caso de uso para ver los productos de un establecimiento.

Caso de uso	Añadir establecimiento
Actor principal	Usuario.
Precondición	El actor está identificado en el sistema con un producto seleccionado.
Disparador	Un usuario quiere añadir un nuevo establecimiento a un producto.
Escenario principal	
1. El actor busca un establecimiento [→Buscar Establecimiento]. 2. El actor selecciona un establecimiento e indica un nuevo precio. 3. El sistema realiza la validación y guarda. 4. El caso de uso termina.	
Extensiones	
2a. Alguno de los datos introducidos por el actor son incorrectos: 2a1. El sistema notifica al actor que los datos introducidos no son correctos.	

2a3. El caso de uso continúa en el paso 2 del escenario principal.

Tabla 12: Añadir un establecimiento al producto.

Caso de uso	Añadir producto
Actor principal	Usuario.
Precondición	El actor está identificado en el sistema con un establecimiento seleccionado.
Disparador	Un usuario quiere añadir un producto.
Escenario principal	
1. El actor indica el producto a añadir. 2. El sistema busca el producto y lo muestra. 3. El actor introduce un precio [→Añadir precio al producto] 4. El sistema añade el producto al establecimiento. 5. El caso de uso termina.	
Extensiones	
2a. El sistema no encuentra el producto: 2a1. El actor quiere crear un nuevo producto con su precio [→Crear producto]. 2a3. El caso de uso termina.	

Tabla 13: Caso de uso para añadir un producto a un establecimiento.

Caso de uso	Crear producto
Actor principal	Usuario.
Precondición	El actor está identificado en el sistema y viene del caso de uso "Añadir producto".
Disparador	Un usuario quiere añadir un producto no existente en el sistema.
Escenario principal	
1. El sistema muestra la información a completar. 2. El actor introduce los datos del producto a crear. 3. El actor añade un nuevo precio [→Añadir precio al producto]. 4. El sistema valida y añade el producto al establecimiento. 5. El caso de uso termina.	
Extensiones	
4a. Alguno de los datos introducidos por el actor son incorrectos: 4a1. El sistema notifica al actor que los datos introducidos no son correctos. 4a3. El caso de uso continúa en el paso 1 del escenario principal.	

Tabla 14: Caso de uso para añadir un producto al sistema.

Caso de uso	Añadir precio al producto.
Actor principal	Usuario
Precondición	El actor está identificado en el sistema. Ha seleccionado un producto y un establecimiento.
Disparador	Un usuario quiere añadir un precio a un producto.
Escenario principal	
1. El actor introduce el nuevo precio. 2. El sistema valida y añade el nuevo precio. 3. El caso de uso termina.	
Extensiones	
2a. Alguno de los datos introducidos por el actor son incorrectos: 2a1. El sistema notifica al actor que los datos introducidos no son correctos. 2a3. El caso de uso continúa en el paso 1 del escenario principal.	

Tabla 15: Caso de uso para añadir un precio al producto.

Caso de uso	Crear nuevo establecimiento.
Actor principal	Usuario
Precondición	El actor está identificado en el sistema.
Disparador	Un usuario quiere introducir un nuevo establecimiento en el sistema.
Escenario principal	
1. El sistema muestra la información a completar. 2. El actor introduce la información solicitada. 3. El sistema valida y añade el nuevo establecimiento. 4. El caso de uso termina.	
Extensiones	
2a. Alguno de los datos introducidos por el actor son incorrectos: 2a1. El sistema notifica al actor que los datos introducidos no son correctos. 2a3. El caso de uso continúa en el paso 1 del escenario principal.	

Tabla 16: Caso de uso para añadir un nuevo establecimiento al sistema.

10.1.3. Diagrama conceptual

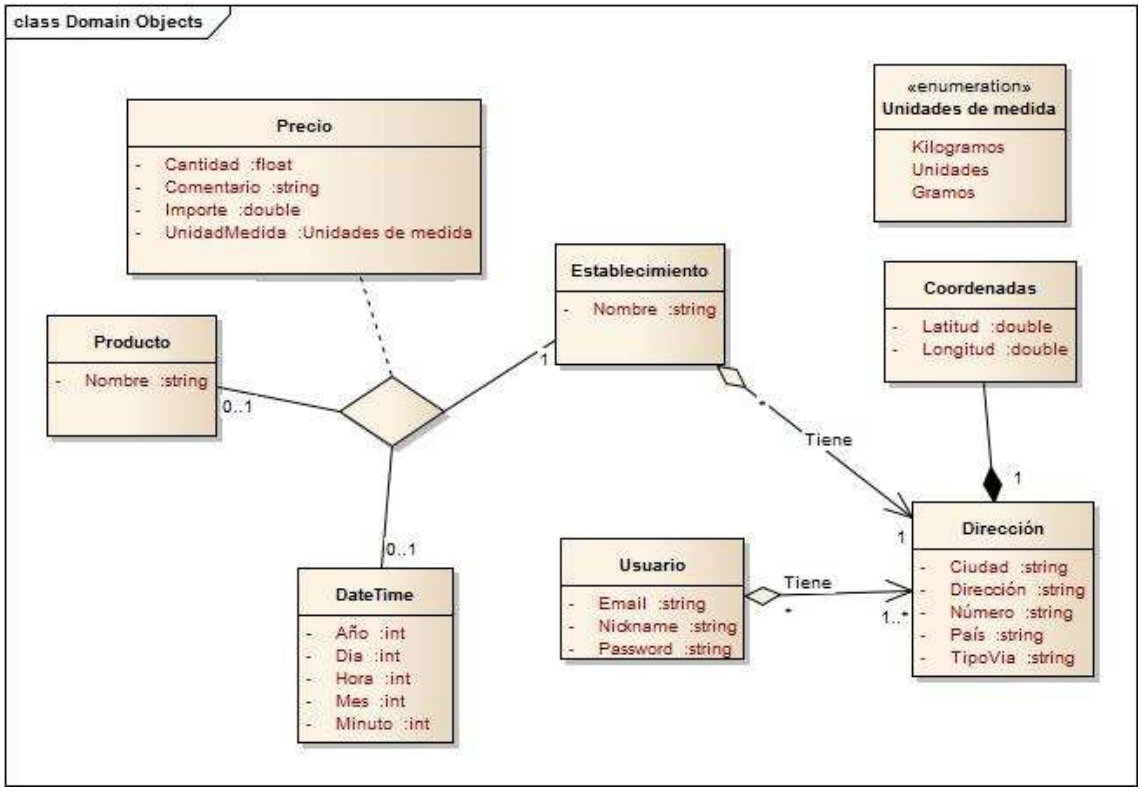


Ilustración 6: Diagrama conceptual.

10.2. Requisitos de calidad (no funcionales)

10.2.1. Requisitos de estilo y apariencia

Requisito	Estilo y apariencia	**Requisito número**	001
Stakeholders	Usuario		
Descripción	Las interfaces han de ser simples e intuitivas.		
Justificación	Con una interfaz de usuario simple se disminuyen las distracciones al usuario y facilitando su rapidez de uso.		
Criterio de satisfacción	Se realizara una encuesta entre una porción de los usuarios elegida al azar en la que se solicitara que valoren el aspecto y apariencia. Se considerara que cumplimos el requisito si más del 70% de los encuestados valora positivamente.		
Satisfacción	4	**Insatisfacción**	2

Prioridad	Media – baja	**Conflicto**	-
Historia	18/04/2014: creación		

Tabla 17: Requisito estilo y apariencia.

10.2.2. Requisitos de usabilidad y humanidad

Requisito	Facilidad de uso	**Requisito número**	002
Stakeholders	Usuario		
Descripción	Las interfaces deben ser fáciles de usar para cualquier usuario. Para ello, tienen que ser claras y las principales funcionalidades deben estar a la vista desde cualquiera parte de la web.		
Justificación	Queremos que el usuario se sienta cómodo al utilizarlo de forma habitual.		
Criterio de satisfacción	Se realizara una encuesta entre una porción de los usuarios elegida al azar en la que se solicitara que valoren el aspecto y apariencia. Se considerara que cumplimos el requisito si más del 70% de los encuestados valora positivamente.		
Satisfacción	2	**Insatisfacción**	4
Prioridad	Media – baja	**Conflicto**	-
Historia	18/04/2014: creación 29/04/2014: Modificación		

Tabla 18: Requisito de facilidad de uso.

Requisito	Idioma	**Requisito número**	003
Stakeholders	Usuario		
Descripción	Las interfaces deben estar al menos en castellano para que pueda ser usada a nivel estatal.		
Justificación	Para atraer el máximo número de usuarios las interfaces deben estar al menos en castellano con opción de poder incluir más idiomas como por ejemplo el catalán, gallego y euskera.		
Criterio de satisfacción	Se realizará un test. Para ello grupos de 50 personas utilizaran la versión en su idioma y contestaran al test al finalizar la prueba. El test preguntará si existen partes mal traducidas o si el usuario se ha perdido al realizar alguna		

	funcionalidad. Se considerará que el requisito se ha satisfecho si el 90% de los encuestados no han encontrado palabras mal traducidas y no se han perdido.		
Satisfacción	3	**Insatisfacción**	3
Prioridad	Alta	**Conflicto**	-
Historia	18/04/2014: creación		

Tabla 19: Requisito del idioma.

Requisito	Iconos	**Requisito número**	004
Stakeholders	Usuario		
Descripción	Las interfaces deberán de tener iconos que sean fácilmente reconocibles.		
Justificación	Los usuarios tienen que interpretar rápidamente la función de cada botón de nuestro sistema y asociarlo con su uso común en los sistemas informáticos actuales.		
Criterio de satisfacción	Se hará un test de velocidad en que un grupo aleatorio de usuarios tendrán que adivinar la funcionalidad de cada icono, y al menos un 90% debe superarlo con éxito.		
Satisfacción	2	**Insatisfacción**	4
Prioridad	Media	**Conflicto**	-
Historia	18/04/2014: creación		

Tabla 20: Requisito de los iconos.

Requisito	Accesibilidad	**Requisito número**	005
Stakeholders	Usuario		
Descripción	El sistema debe ser accedido vía dispositivos móviles, ya sea mediante aplicaciones específicas, o adaptando la página web a formatos para móvil.		
Justificación	Los usuarios tienen que poder acceder al servicio siempre que lo necesiten para facilitar la actualización de precios.		
Criterio de satisfacción	En el 99,9% de los casos, los usuarios del sistema podrán acceder a la web y a todos sus servicios, vía navegador o vía aplicación para móvil.		
Satisfacción	3	**Insatisfacción**	4

Prioridad	Media	**Conflicto**	-
Historia	18/04/2014: creación		

Tabla 21: Requisito de accesibilidad.

10.2.3. Requisitos de rendimiento

Requisito	Velocidad y latencia	**Requisito número**	006
Stakeholders	Usuario		
Descripción	El sistema debe mostrar el contenido de cualquier sección en no más de tres segundos.		
Justificación	Queremos que el usuario no tenga que esperar mucho cuando esté realizando gestiones.		
Criterio de satisfacción	Se realizara un test aleatorio y automático de acceso a diferentes secciones durante distintas zonas del día. Se considerará que se ha satisfecho el requisito si el 85% de los accesos se hayan hecho antes de tres segundos.		
Satisfacción	2	**Insatisfacción**	4
Prioridad	Media	**Conflicto**	-
Historia	18/04/2014: creación		

Tabla 22: Requisito de velocidad y latencia.

Requisito	Capacidad en número de usuarios	**Requisito número**	007
Stakeholders	Cliente.		
Descripción	El sistema debe de poder soportar un gran número de usuarios conectados al mismo tiempo.		
Justificación	Los usuarios deben de poder utilizar todas las funcionalidades del sistema independientemente del número de usuarios conectados en un momento determinado.		
Criterio de satisfacción	El sistema tiene que poder soportar un volumen alto de usuarios sin verse perjudicado su rendimiento ni estabilidad. A medida que el número de usuarios vaya creciendo se irá escalando el		

	sistema.		
Satisfacción	1	**Insatisfacción**	5
Prioridad	Media	**Conflicto**	-
Historia	18/04/2014: creación		

Tabla 23: Requisito de la capacidad en número de usuarios.

Requisito	Disponibilidad	**Requisito número**	008
Stakeholders	Cliente		
Descripción	El sistema debe de estar disponible el mayor tiempo posible.		
Justificación	Los usuarios quieres poder acceder al sistema cuando quieran.		
Criterio de satisfacción	El sistema debe de estar disponible con todas sus funcionalidades el 99% del tiempo, solo puede estar inoperativo por la noche por tareas de mantenimiento si se requieren.		
Satisfacción	1	**Insatisfacción**	5
Prioridad	Alta	**Conflicto**	-
Historia	18/04/2014: creación		

Tabla 24: Requisito de la disponibilidad.

Requisito	Escalabilidad	**Requisito número**	009
Stakeholders	Cliente		
Descripción	El sistema debe de poder crecer en un futuro, aumentando sus recursos para ofrecer capacidad para un mayor número de usuarios, así como para soportar un mayor número de transacciones.		
Justificación	El sistema se debe de poder ajustar a un futuro crecimiento		
Criterio de satisfacción	El sistema podrá crecer tanto como sea necesario para poder soportar cualquier volumen de usuarios, así como permitir que se realicen cualquier número de transacciones. A medida que el número de usuarios vaya creciendo se irá escalando el sistema.		
Satisfacción	3	**Insatisfacción**	3

Prioridad	Media	**Conflicto**	-
Historia	18/04/2014: creación		

Tabla 25: Requisito de escalabilidad.

10.2.4. Requisitos de seguridad

Requisito	Seguridad	**Requisito número**	010
Stakeholders	Cliente		
Descripción	Se debe de realizar una copia de seguridad de los datos cada doce horas.		
Justificación	Para evitar posibles fallos en el sistema en un futuro y/o poder restablecer el sistema en un estado anterior, es necesario que cada 12 horas se realice una copia de seguridad de los datos almacenados.		
Criterio de satisfacción	El sistema realizará un uso normal durante dos semanas con usuarios en activo. Una vez terminadas estas dos semanas se encogerán 9 copias de seguridad al azar de las 28 realizadas y se comprobará que el sistema puede recuperar en un estado anterior y este es funcional. Se considerará que se ha satisfecho el requisito si el 80% de los casos pueden recuperar y operar con normalidad el sistema.		
Satisfacción	2	**Insatisfacción**	4
Prioridad	Media -Alta	**Conflicto**	-
Historia	18/04/2014: creación		

Tabla 26: Requisito de seguridad.

10.2.5. Requisitos operativos

Requisito	Actualizaciones sin errores	**Requisito número**	011
Stakeholders	Cliente		
Descripción	Siempre que haya una actualización en el sistema se debe asegurar que no hay errores graves surgidos de la actualización.		
Justificación	Las actualizaciones del sistema son para mejorar las prestaciones, con el objetivo de mejorar los servicios y obtener más usuarios, si se producen errores es posible que el prestigio caiga y los usuarios estén descontentos.		

Criterio de satisfacción	Después de cada actualización se realizarán test automáticos donde se probarán todas las funciones básicas del sistema. El 95% de las actualizaciones no deben dar ningún error para poder considerar que se cumple el requisito. En caso de que haya error se solucionará en un máximo de 24 horas.		
Satisfacción	3	**Insatisfacción**	5
Prioridad	Media - Alta	**Conflicto**	-
Historia	18/04/2014: creación		

Tabla 27: Requisito de actualizaciones sin errores.

10.2.6. Requisitos de actuación

Requisito	Privacidad	**Requisito número**	013
Stakeholders	Usuario		
Descripción	Los usuarios deben de estar protegidos y mantenerse en el anonimato.		
Justificación	Los usuarios quieren proteger sus datos personales y mantener su privacidad en el sistema ya que no son requeridos.		
Criterio de satisfacción	El sistema no debe de mostrar ningún tipo de información de un usuario al resto excepto su alias o nickname.		
Satisfacción	4	**Insatisfacción**	5
Prioridad	Alta	**Conflicto**	-
Historia	21/04/2014: creación		

Tabla 28: Requisito de privacidad.

11. Arquitectura del sistema (7)

En este apartado se define la arquitectura a utilizar. Todo el sistema se implementara con lenguaje C#.

11.1. Arquitectura N-Capas con Orientación al dominio

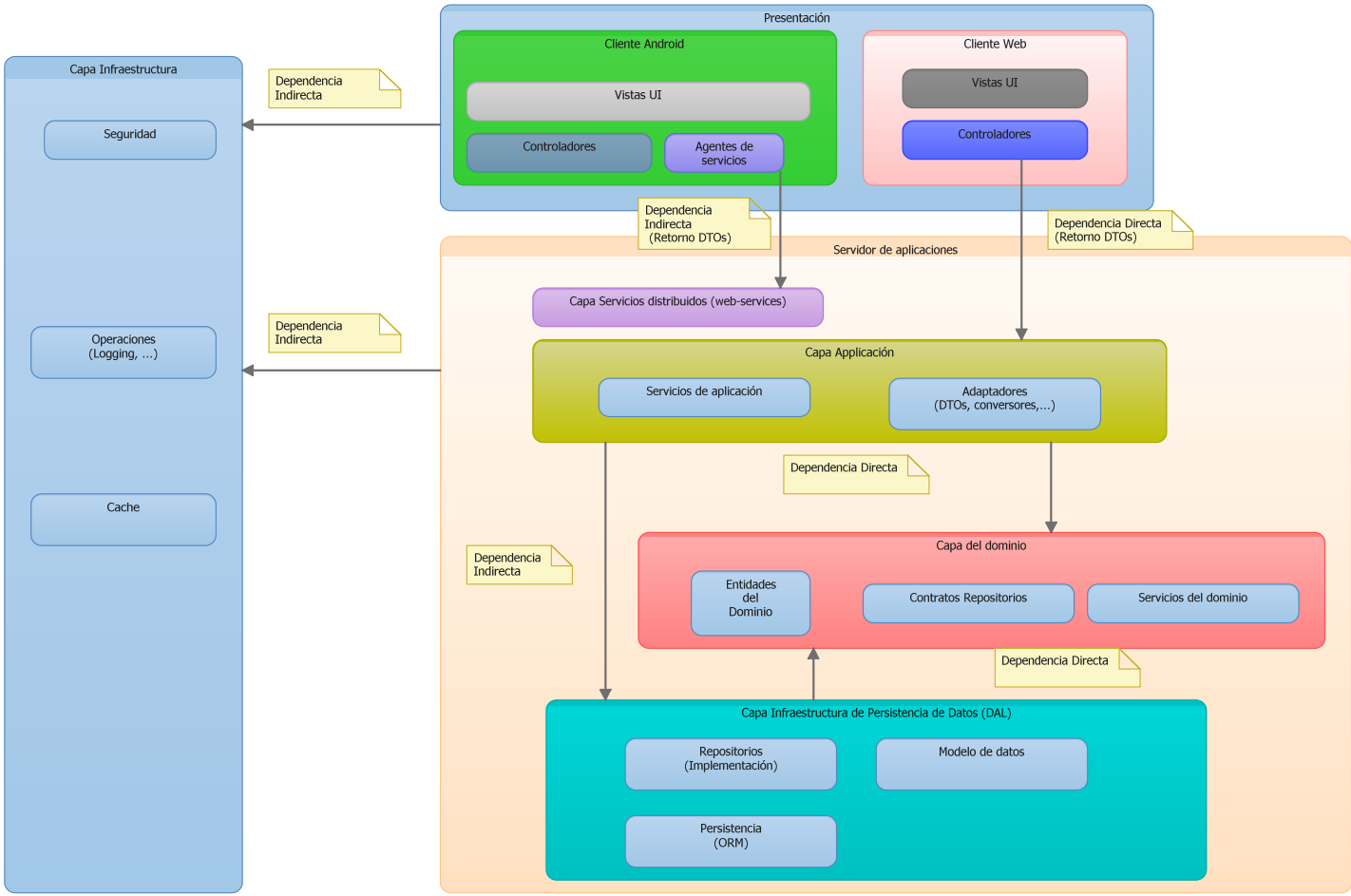


Ilustración 7: Arquitectura N-capas con orientación al dominio.

11.1.1. Descripción de las capas

Capa de presentación

Cliente Android

Aplicación Android para utilizar e interactuar en la comunidad.

> **Vistas UI**: Subcapa de los componentes visuales de la aplicación para dispositivos Android.
> **Controladores**: Subcapa de los componentes controladores de la capa vistas.
> **Agentes de servicios**: Subcapa de los componentes necesarios para conectarse al servicio externo (Capa de servicios distribuidos).

Cliente Web

Aplicación web para utilizar e interactuar en la comunidad.

> **Vistas UI:** Subcapa de los componentes visuales de la aplicación web.
> **Controladores**: Subcapa de los componentes controladores de la capa vistas.

Capa servidor de aplicaciones

Esta capa engloba el ecosistema propio de la aplicación que está dividida en 4 capas.

Capa de servicios distribuidos

Capa que se encarga de ofrecer un servicio web con el que las aplicaciones cliente poder interactuar con la comunidad.

Capa de aplicación

> **Servicios de Aplicación:** Subcapa que contiene los componentes necesarios para realizar las tareas y coordinadores de los cacos de uso.
> **Adaptadores:** contienes las clases DTOs y conversores de objetos.

Capa del dominio

> **Entidades del dominio:** Contiene las clases (entidades) del dominio.
> **Contratos de repositorios:** Contiene los contratos (interfaces) de los repositorios de acceso a datos.
> **Servicios del dominio:** Contiene los componentes necesarios que engloban la lógica de negocio en la que intervienen varias entidades del dominio.

Capa de infraestructura de persistencia de datos (DAL)

> **Modelo de datos:** Contiene el modelo de datos de persistencia.
> **Implementación de repositorios:** Contiene la implementación de los contratos (interfaces) de los repositorios de acceso a datos de la capa de dominio.
> **Persistencia:** Infraestructura tecnológica del ORM.

11.2. Patrones utilizados

- **Arquitectura 2-Tier:** Arquitectura que describe la división física en 2. Se establece una separación de física de cliente-servidor.
- **Arquitectura N-Capas (7):** Arquitectura que describe la división lógica de componentes y funcionalidades.
- **Arquitectura basada en la orientación del dominio (DDD) (7):** Se basa en conseguir un modelo orientado a objetos que refleja el conocimiento de un dominio dado y que sea completamente independiente de cualquier concepto de comunicación o infraestructura.
- **Inversión de control (IoC):** Consiste en invertir la responsabilidad de los componentes. Este patrón ayuda a conseguir un mayor desacoplamiento.
- **Inyección de dependencias (DI):** Consiste en inyectar las dependencias de los objetos de forma dinámica.
- **Repositorio:** Consiste en encapsular la lógica requerida para acceder a la fuente de datos.
- **Service Locator:** Consiste en centralizar la gestión de servicios en un único lugar. Utilizado en los patrones de inversión de control (IoC) e inyección de dependencias (DI).
- **Singleton:** Una clase tiene una única instancia.
- **Factoría:** Consiste en centralizar la creación e inicialización en un único lugar. Utilizado en los patrones de inversión de control (IoC) e inyección de dependencias (DI).
- **Lazy Initialization:** Consiste en retrasar la inicialización de las propiedades en los objetos hasta que no se necesiten. En el momento en el que son necesarios es cuando se realiza la inicialización.
- **Super-type:** Engloba en clases base funcionalidades comunes de varias clases.

11.3. Capa de dominio

11.3.1. Diagrama de clases

El diagrama de clases ha salido del diagrama conceptual del punto 0.

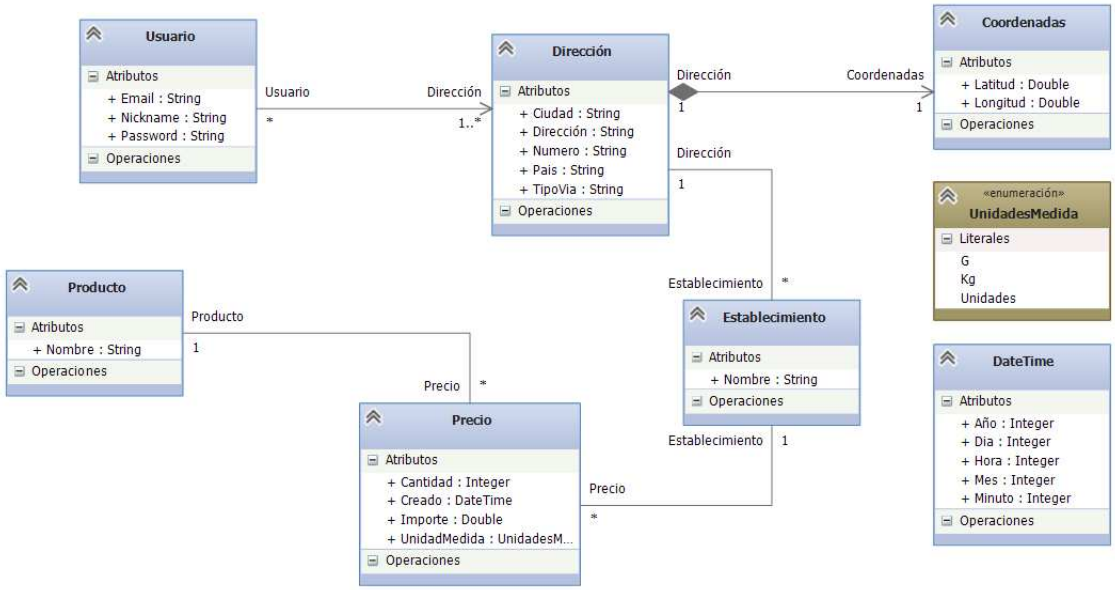


Ilustración 8: Capa del dominio - Diagrama de clases.

Las asociaciones sin punta de flecha indican asociaciones bidireccionales.

11.3.2. Contratos de los repositorios

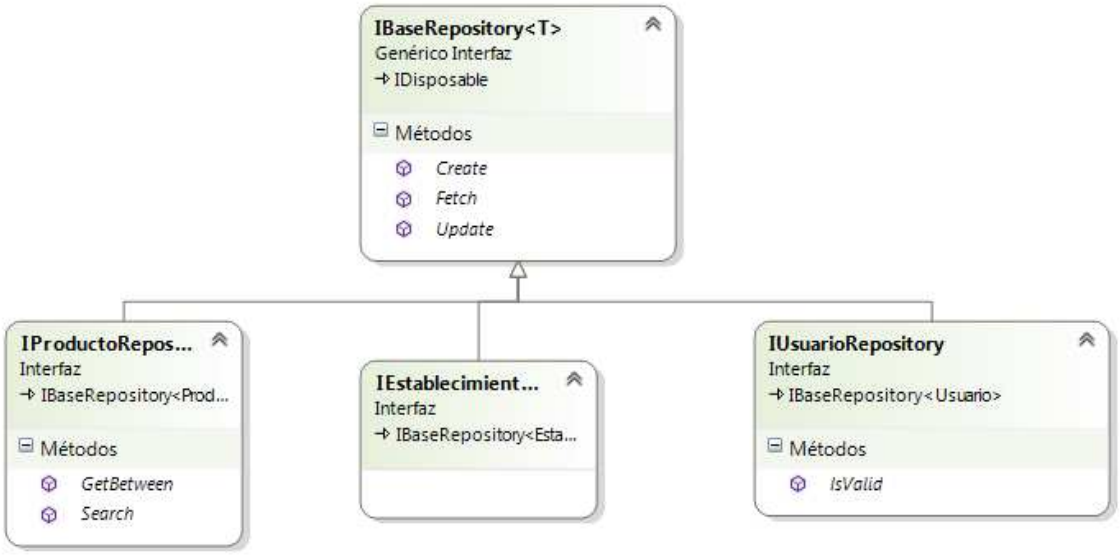


Ilustración 9: Capa del dominio - Diagrama de contratos de los repositorios.

Nombre	IBaseRepository<T>
Tipo	Interfaz.
Padre	-
Descripción	Interfaz de tipo genérica para el acceso a datos.
Métodos	
Nombre	**Descripción**
Void Create(T entity)	Crear un nuevo objeto en la base de datos de tipo T.
Void Update(T entity)	Actualiza un objeto en las base de datos de tipo T.
T Fecth(DataFetchRequest fetch)	Busca un objeto en la base de datos de tipo T.

Tabla 29: Interfaz IBaseRepository<T>.

Nombre	IProductoRepository
Tipo	Interfaz.
Padre	IBaseRepository<Producto>
Descripción	Define la interfaz de acceso a datos de la entidad “Producto”.
Métodos	
Nombre	**Descripción**
IEnumerable<Precio> GetBetween(string nombre, double latSup, double lonSup, double latInf, double lonInf)	Obtiene los precios de un producto en una ventana geográfica especificada. **Parámetros**: • Nombre: Nombre del producto. • latSup: Latitud superior. • lonSup: Longitud superior. • latInf: Latitud inferior. • lonInf: Longitud inferior.
IEnumerable<Producto> Search(string term);	Busca los productos que coincidan con el parámetro de entrada “term”.

Tabla 30: Interfaz IProductoRepository.

Nombre	IEstablecimientoRepository
Tipo	Interfaz.
Padre	IBaseRepository<Establecimiento>
Descripción	Define la interfaz de acceso a datos de la entidad “Establecimiento”.
Métodos	
Nombre	**Descripción**
-	-

Tabla 31: Interfaz IEstablecimientoRepository.

Nombre	IUsuarioRepository
Tipo	Interfaz.
Padre	IBaseRepository<Usuario>
Descripción	Define la interfaz de acceso a datos de la entidad “Usuario”.
Métodos	

Nombre	**Descripción**
bool IsValid(string nickName, string password);	Comprueba si un usuario es válido introduciendo su nickname y contraseña encriptada.

Tabla 32: Interfaz IUsuarioRepository.

11.3.3. Capa de servicios

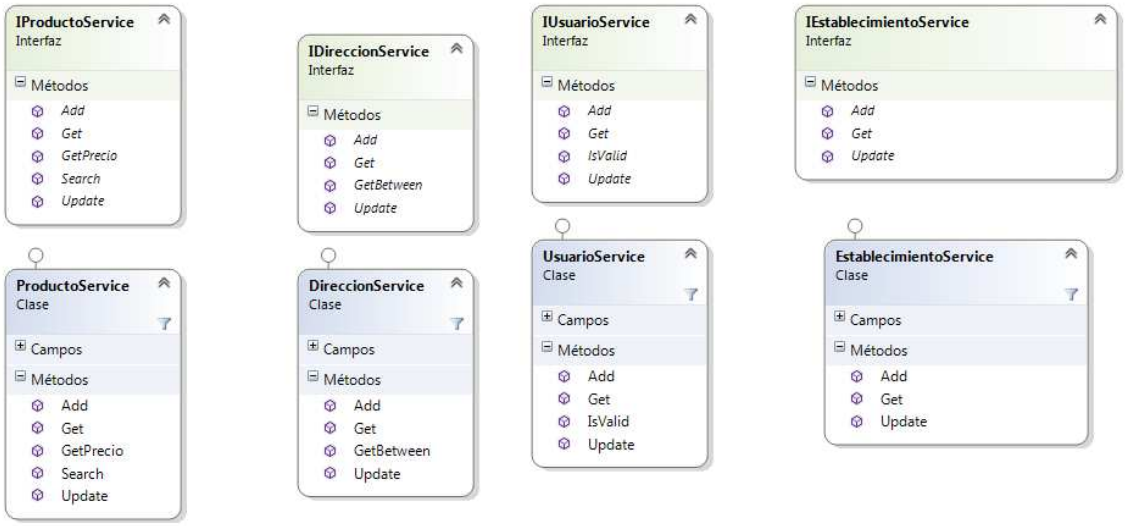


Ilustración 10: Capa del dominio - Diagrama de servicios.

Esta capa se encarga de realizar las operaciones del dominio teniendo en cuentas las reglas de integración.

Nombre	IProductoService.
Tipo	Interfaz.
Padre	-
Descripción	Define la interfaz para realizar operaciones del dominio de la entidad “Producto”.
Métodos	

Nombre	**Descripción**
void Add(Producto producto);	Añade un nuevo producto al dominio.
Producto Get(string name);	Obtiene el producto que coincida su nombre con el parámetro de entrada.
IEnumerable<Producto> Search(string name);	Obtiene una lista de productos que coincida su nombre total o parcialmente con el parámetro de entrada.

void Update(Producto producto);	Actualiza un producto del dominio.
IEnumerable<Precio> GetPrecios(string name, double latitud, double longitud, double distancia);	Obtiene los precios de un producto en un área geográfica indicando su nombre, una latitud y longitud como punto central y una distancia máxima en metros.

Tabla 33: Interfaz IProductoService.

Nombre	IDireccionService.
Tipo	Interfaz.
Padre	-
Descripción	Define la interfaz para realizar operaciones del dominio de la entidad "Direccion".
Métodos	
Nombre	**Descripción**
void Add(Direccion direccion);	Añade una nueva dirección al dominio.
Direccion Get(double lat, double lon);	Obtiene a dirección que coincida con la latitud y longitud introducidas.
void Update(Direccion direccion);	Actualiza una dirección del dominio.
IEnumerable<Direccion> GetBetween(double lat, double lon, double distancia);	Obtiene las direcciones del dominio en un área geográfica indicando una latitud y longitud como punto central y una distancia máxima en metros.

Tabla 34: Interfaz IDireccionService.

Nombre	IUsuarioService.
Tipo	Interfaz.
Padre	-
Descripción	Define la interfaz para realizar operaciones del dominio de la entidad "Usuario".
Métodos	
Nombre	**Descripción**
void Add(Usuario usuario);	Añade un nuevo usuario al dominio.
Usuario Get(string nickname);	Obtiene un usuario que coincida su nickname con el parámetro de entrada.
void Update(Usuario usuario);	Actualiza un usuario del dominio.
bool IsValid(string nickName, string password);	Comprueba si un usuario es válido introduciendo su nickname y contraseña encriptada.

Tabla 35: Interfaz IUsuarioService.

Nombre	IEstablecimientoService.
Tipo	Interfaz.
Padre	-
Descripción	Define la interfaz para realizar operaciones del dominio de la entidad “Establecimiento”.
Métodos	
Nombre	**Descripción**
void Add(Establecimiento establecimiento);	Añade un nuevo establecimiento al dominio.
Establecimiento Get(Guid id);	Obtiene un establecimiento a partir de su identificador.
void Update(Establecimiento establecimiento);	Actualiza un establecimiento del dominio.

Tabla 36: Interfaz IEstablecimientoService.

Nombre	ProductoService.
Tipo	Clase
Padre	IProductoService.
Descripción	Implementación del padre.

Tabla 37: clase ProductoService.

Nombre	DireccionService.
Tipo	Clase
Padre	IDireccionService.
Descripción	Implementación del padre.

Tabla 38: clase DireccionService.

Nombre	UsuarioService.
Tipo	Clase
Padre	IUsuarioService
Descripción	Implementación del padre.

Tabla 39: Clase UsuarioService.

Nombre	EstablecimientoService.
Tipo	Clase
Padre	IEstablecimientoService
Descripción	Implementación del padre.

Tabla 40: Clase EstablecimientoService.

11.4. Capa de infraestructura de datos (DAL)

11.4.1. Modelo de datos

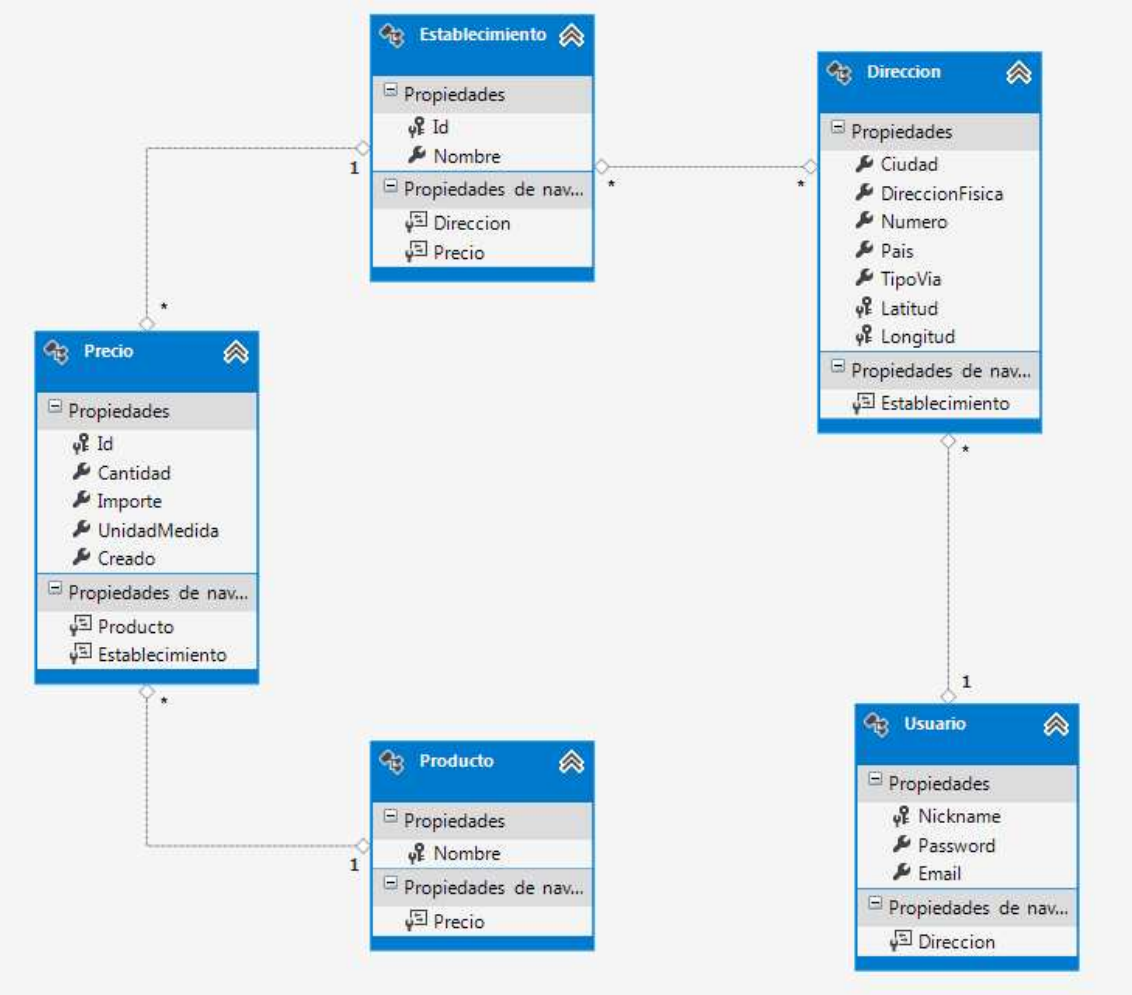


Ilustración 11: Diagrama del modelo de datos.

11.4.2. Implementación de repositorios

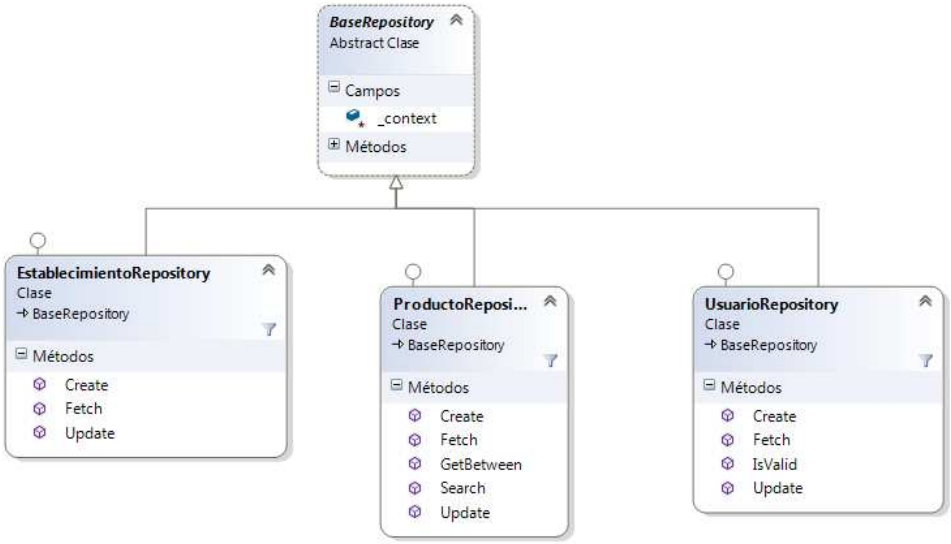


Tabla 41: Diagrama de implementación de los repositorios.

Nombre	BaseRepository
Tipo	Clase
Padre	-
Descripción	Clase de super tipo para el acceso a datos.

Tabla 42: Clase BaseRepository.

Nombre	ProductoRepository
Tipo	Clase
Padre	IProductoRepository
Descripción	Implementación del padre. Ver punto 11.3.2

Tabla 43: Clase ProductoRepository.

Nombre	EstablecimientoRepository
Tipo	Clase
Padre	IEstablecimientoRepository
Descripción	Implementación del padre. Ver punto 11.3.2

Tabla 44: Clase EstablecimientoRepository.

Nombre	UsuarioRepository
Tipo	Clase
Padre	IUsuarioRepository

Descripción	Implementación del padre. Ver punto 11.3.2

Tabla 45: Clase UsuarioRepository.

11.5. Capa de aplicación

11.5.1. Capa de servicios

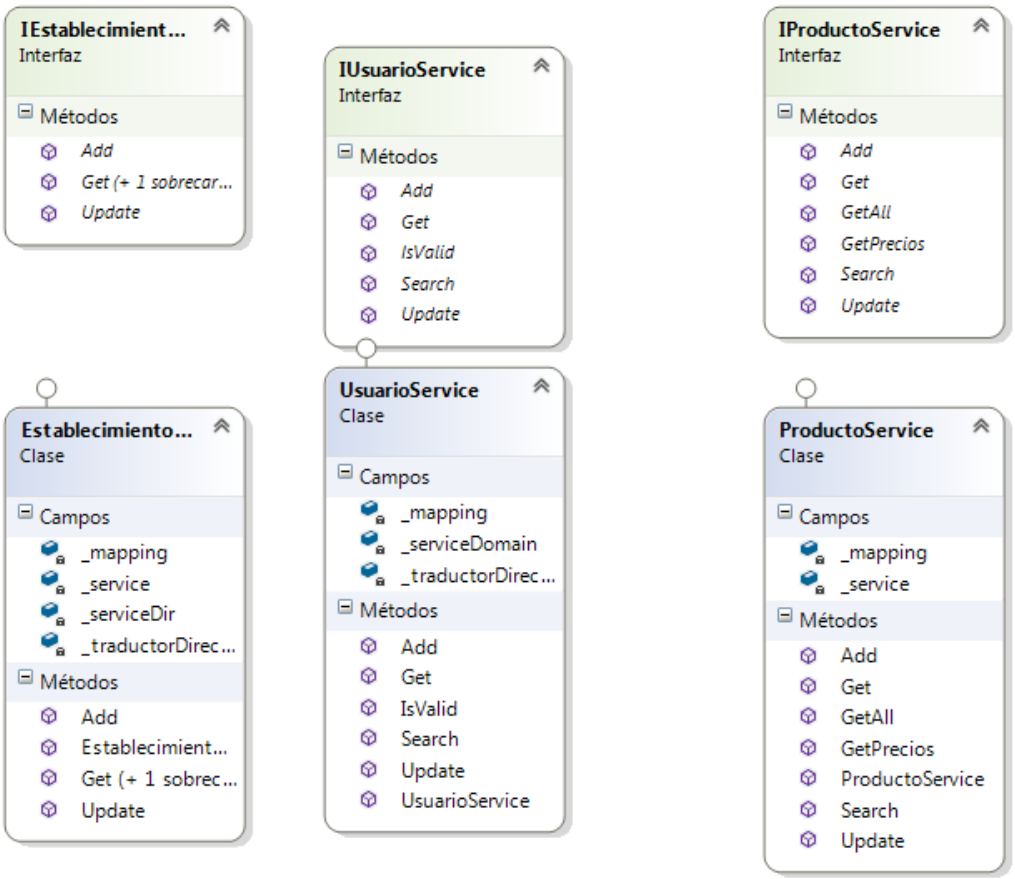


Ilustración 12: Diagrama de la subcapa de servicios de la capa aplicación.

Nombre	IEstablecimientoService.
Tipo	Interfaz.
Padre	-
Descripción	Define la interfaz para realizar operaciones del dominio de la entidad "Establecimiento" con DTOs.
Métodos	
Nombre	**Descripción**
void Add(EstablecimientoDTO establecimiento);	Añade un nuevo establecimiento al dominio.

EstablecimientoDTO Get(string id);	Obtiene un establecimiento a partir de su identificador.
void Update(EstablecimientoDTO establecimiento);	Actualiza un establecimiento del dominio.
IEnumerable< EstablecimientoDTO > Get(double lat, double lon, double distancia);	Obtiene los establecimientos del dominio en un área geográfica indicando una latitud y longitud como punto central y una distancia máxima en metros.

Tabla 46: Interfaz IEstablecimientoService capa de aplicación.

Nombre	IUsuarioService.
Tipo	Interfaz.
Padre	-
Descripción	Define la interfaz para realizar operaciones del dominio de la entidad "Usuario" con DTOs.
Métodos	
Nombre	**Descripción**
void Add(UsuarioDTO usuario);	Añade un nuevo usuario al dominio.
Usuario Get(string nickname);	Obtiene un usuario que coincida su nickname con el parámetro de entrada.
void Update(UsuarioDTO usuario);	Actualiza un usuario del dominio.
bool IsValid(string nickName, string password);	Comprueba si un usuario es válido introduciendo su nickname y contraseña encriptada.
void AddDireccion(UsuarioDTO usuario);	Añade direcciones al usuario. El parámetro de entrada debe contener las direcciones a añadir.
void UpdateDireccion(UsuarioDTO usuario);	Actualiza las direcciones del usuario. El parámetro de entrada debe contener las direcciones a actualizar. Actualizar una dirección significa desvincular la dirección con que coincida con la latitud y longitud de la dirección y asignarle una nueva dirección con el resto de parámetros de la dirección.
void DeleteDireccion(UsuarioDTO usuario);	Desvincula las direcciones del usuario. El parámetro de entrada debe contener las direcciones a desvincular.

Tabla 47: Interfaz IUsuarioService capa de aplicación.

Nombre	IProductoService.
Tipo	Interfaz.
Padre	-
Descripción	Define la interfaz para realizar operaciones del dominio de la

	entidad “Producto” con DTOs.
Métodos	
Nombre	**Descripción**
void Add(ProductoDTO producto);	Añade un nuevo producto al dominio.
ProductoDTO Get(string name);	Obtiene el producto que coincida su nombre con el parámetro de entrada.
IEnumerable<ProductoDTO> Search(string name);	Obtiene una lista de productos que coincida su nombre total o parcialmente con el parámetro de entrada.
void Update(ProductoDTO producto);	Actualiza un producto del dominio.
IEnumerable<PrecioDTO> GetPrecios(string name, double latitud, double longitud, double distancia);	Obtiene los precios de un producto en un área geográfica indicando su nombre, una latitud y longitud como punto central y una distancia máxima en metros.

Tabla 48: Interfaz IProductoService capa de aplicación.

Nombre	EstablecimientoService
Tipo	Clase
Padre	IEstablecimientoService
Descripción	Implementación del padre de la capa de aplicación. Ver punto 11.5.1

Tabla 49: Clase EstablecimientoService capa de aplicación.

Nombre	ProductoService
Tipo	Clase
Padre	IProductoService
Descripción	Implementación del padre de la capa de aplicación. Ver punto 11.5.1

Tabla 50: Clase ProductoService capa de aplicación.

Nombre	UsuarioService
Tipo	Clase
Padre	IUsuarioService
Descripción	Implementación del padre de la capa de aplicación. Ver punto 11.5.1

Tabla 51: Clase UsuarioService capa de aplicación.

11.5.2. Capa de DTOs

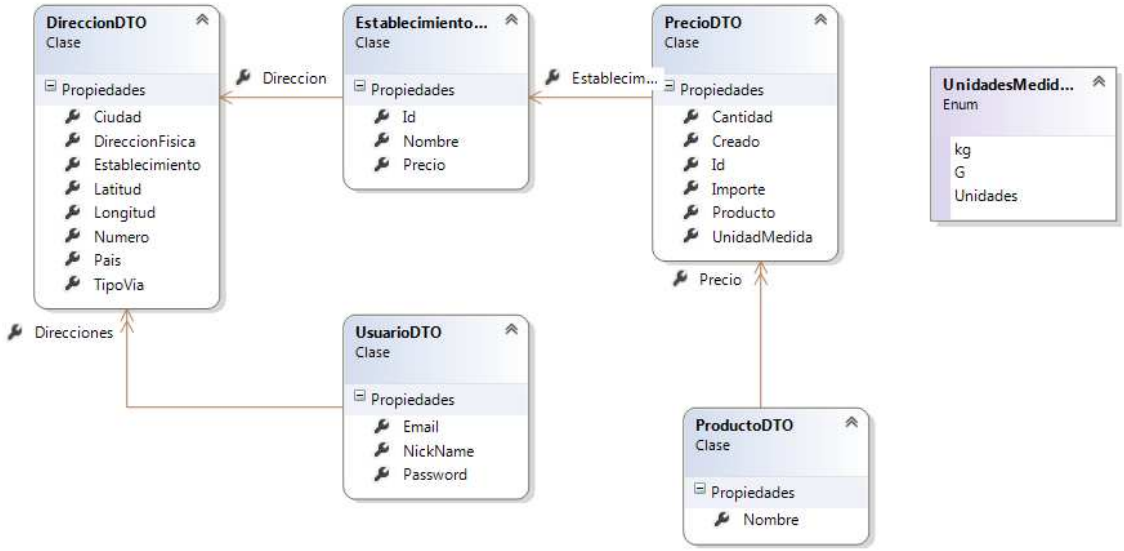
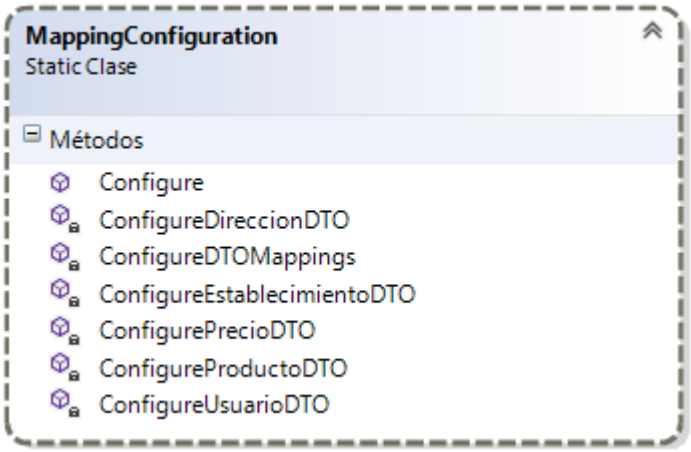


Ilustración 13: Diagrama de DTOs.

Objetos de transferencia de datos. Es utilizado para transportar datos entre procesos, en este caso entre las capas de presentación y el sistema.

11.5.3. Capa de adaptadores



Configura el mapeo entre entidades del dominio y clases DTO.

11.5.4. Capa de agentes

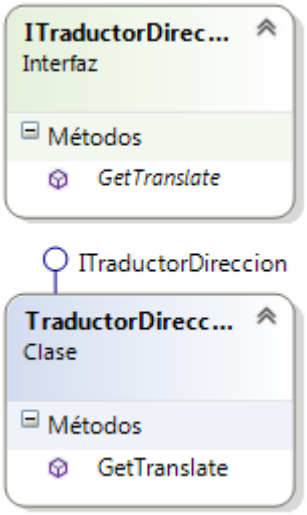


Ilustración 14: Traductor de direcciones a coordenadas geográficas.

Nombre	ITraductorDireccion.
Tipo	Interfaz.
Padre	-
Descripción	Define la interfaz para realizar la traducción de direcciones a coordenadas geográficas.
Métodos	
Nombre	**Descripción**
Result GetTranslate(string direccion)	Devuelve la traducción a posición geográfica la dirección introducida a traducir en formato: tipoVia dirección número población.

Tabla 52: Interfaz ITraductorDireccion.

Nombre	TraductorDireccion
Tipo	Clase
Padre	ITraductorDireccion
Descripción	Implementación del padre. Ver punto tabla anterior.

Tabla 53: Clase TraductorDireccion.

11.6. Capa de infraestructura

11.6.1. Capa de comunicaciones

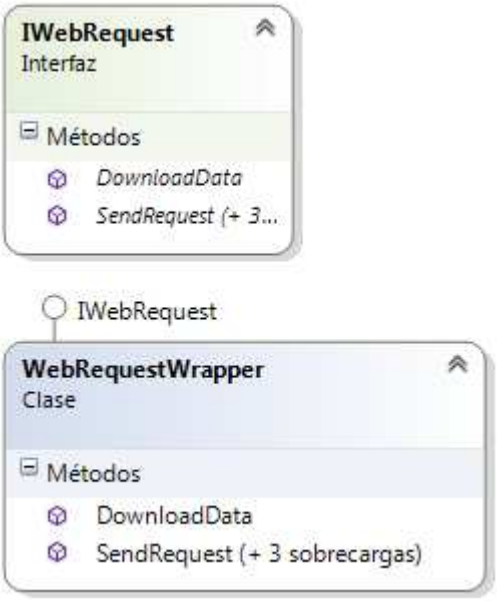


Ilustración 15: capa de comunicaciones de la infraestructura.

Nombre	IWebRequest.
Tipo	Interfaz.
Padre	-
Descripción	Define la interfaz para realizar comunicaciones http.
Métodos	

Nombre	**Descripción**
string SendRequest(string url, object postData);	Realizar una comunicación con la url indicada. Si el parámetro postData tiene contenido se realizara una petición por POST con los datos de dicho parámetro.
string SendRequest(string url, object postData, int? timeout);	Realizar una comunicación con la url indicada con un tiempo máximo indicado en el parámetro timeout. Si el parámetro postData tiene contenido se realizara una petición por POST con los datos de dicho parámetro.
string SendRequest(string url, object postData, string contentType, int? timeout);	Realizar una comunicación con la url indicada con un tiempo máximo indicado en el parámetro timeout y un contentType específico. Si el parámetro postData tiene contenido se realizara una petición por POST con los datos de dicho parámetro.
byte[] DownloadData(string url);	Descarga el contenido de la url indicada.

Tabla 54: Interfaz IWebRequest.

Nombre	WebRequestWrapper
Tipo	Clase
Padre	IWebRequest.
Descripción	Implementación del padre. Ver tabla anterior.

Tabla 55: Clase WebRequestWrapper.

11.6.2. Capa de mapas geográficos

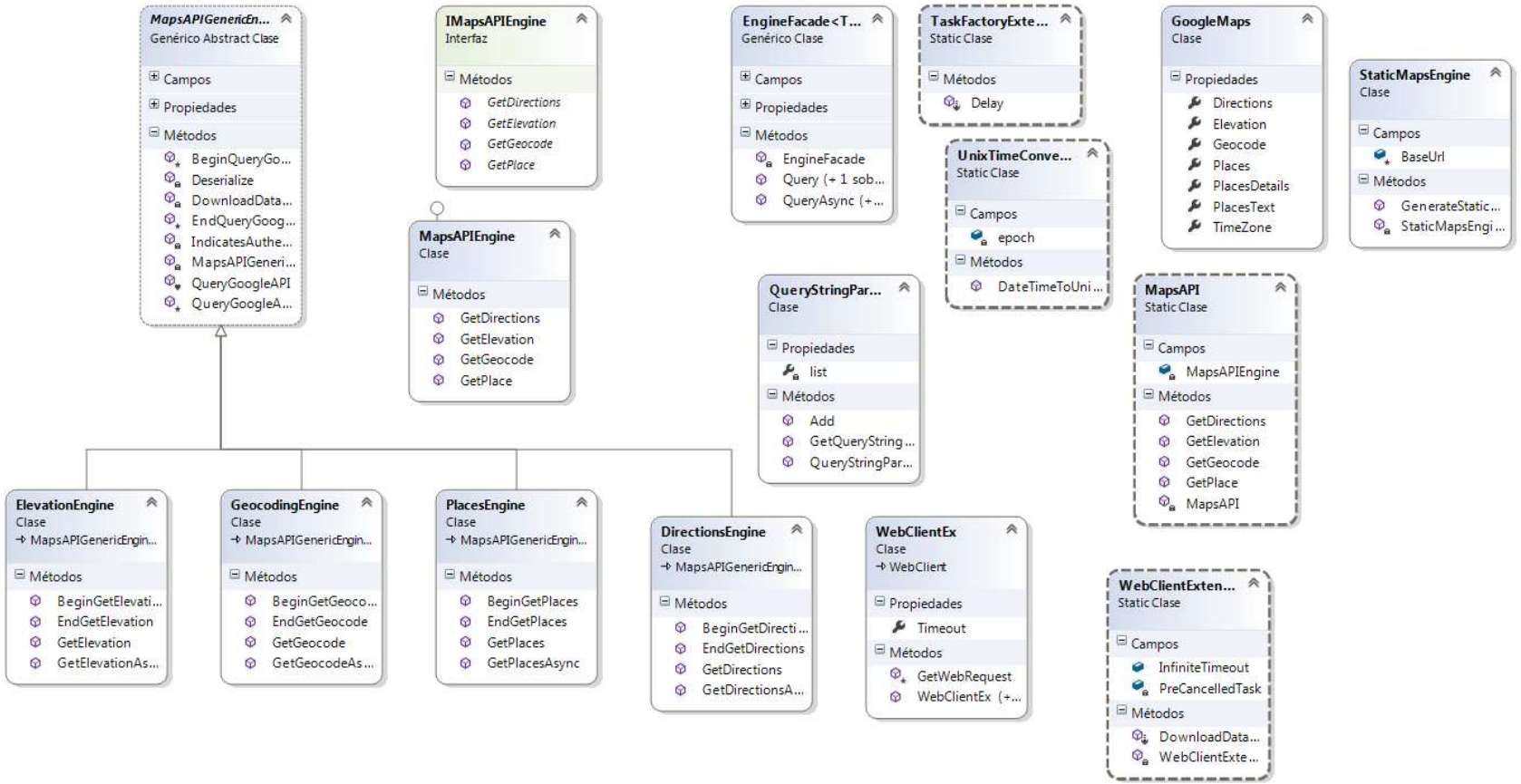


Ilustración 16: Diagrama de la capa de mapas geográficos.

Esta capa se reúne una serie de componentes para realizar una conexión a un servicio de mapas. En el caso de este proyecto, se utilizara para obtener las coordenadas geográficas de una dirección por lo tanto solo se detallara la descripción de las operaciones principales.

Nombre	GoogleMaps
Tipo	Clase estática.
Padre	-
Descripción	Punto de acceso para realizar operaciones.
Propiedades estáticas	

Nombre	**Descripción**
Geocode	Realizar operaciones de geocodificación. Por ejemplo para obtener una posición geográfica de una dirección
Directions	Realizara operaciones con las direcciones.
Elevation	Realizar operaciones de elevación
Places	Realizar operaciones de sitios.
PlacesText	Realizar operaciones con el texto de los sitios.
TimeZone	Realizar operaciones con la zona horaria.
PlacesDetails	Realizar operaciones con los detalles de los sitios.

Tabla 56: Clase estática GoogleMaps.

Nombre	GeocodingEngine
Tipo	Clase.
Padre	MapsAPIGenericEngine<GeocodingRequest, GeocodingResponse>
Descripción	Realizar comunicaciones con el servicio de mapas para geocodificación.
Métodos	

Nombre	**Descripción**
Task<GeocodingResponse> GetGeocodeAsync(GeocodingRequest request)	Realizar una petición asíncrona.
GeocodingResponse GetGeocode(GeocodingRequest request)	Realizar una petición síncrona.

Tabla 57: Clase GeocodingEngine.

11.6.3. Capa de adaptadores

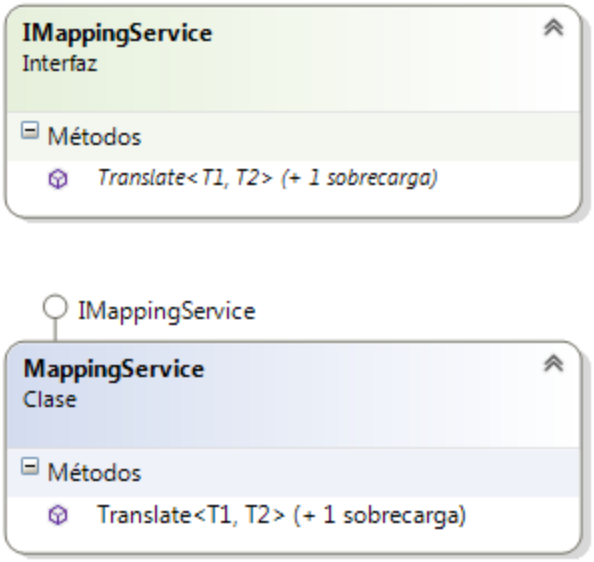


Ilustración 17: Diagrama de la capa de adaptadores capa de infraestructura.

Esta capa define de forma abstracta el componente que adaptara (traducirá) y mapeará la conversión de un tipo de objeto a otro. La configuración del adaptador se realiza en aquellas capas donde se utilicé.

Nombre	IMappingService
Tipo	Interfaz
Padre	-
Descripción	Define la interfaz para realizar la conversión entre dos objetos.
Métodos	
Nombre	**Descripción**
T2 Translate<T1, T2>(T1 source)	Realizar una conversión del objeto de tipo T1 a un objeto de tipo T2.
void Translate<T1, T2>(T1 source, T2 destination)	Realizar una conversión del objeto de tipo T1 a un objeto de tipo T2.

Tabla 58: Interfaz IMappingService.

Nombre	MappingService
Tipo	Clase
Padre	IMappingService.
Descripción	Implementación del padre. Ver tabla anterior.

Tabla 59: Clase MappingService.

11.6.4. Capa de seguridad

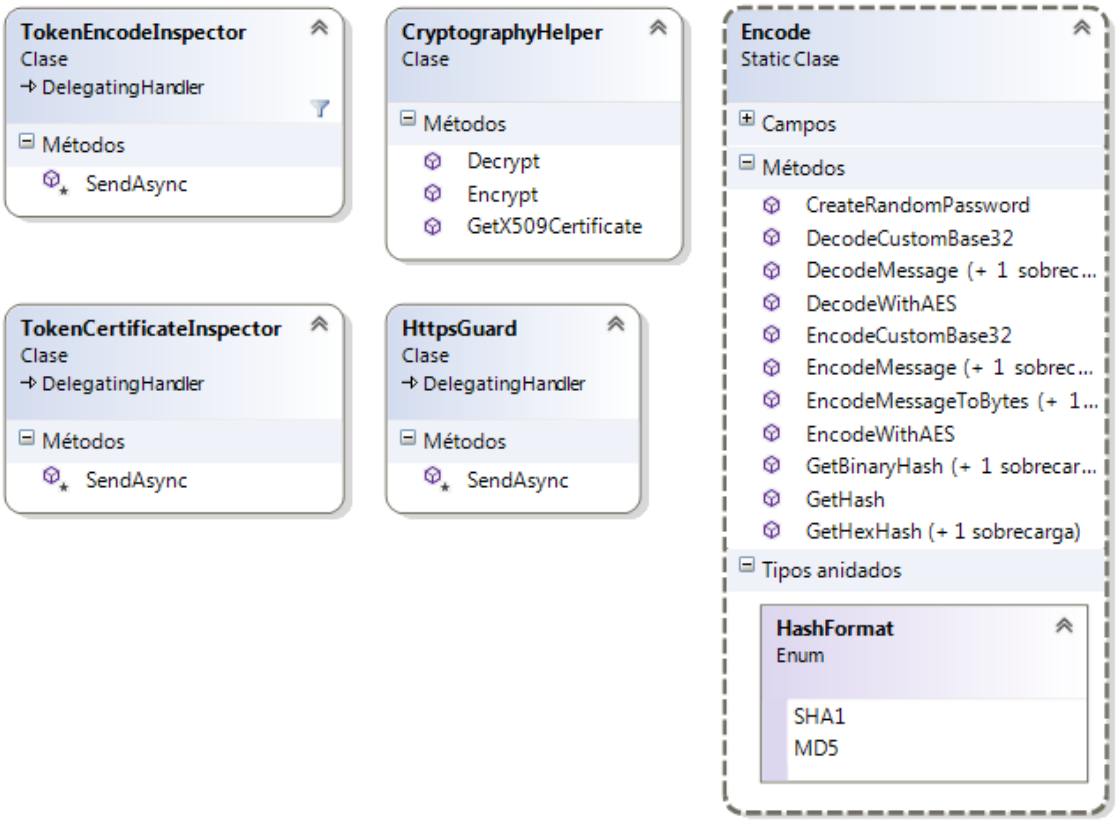


Ilustración 18: Diagrama capa de seguridad.

Esta capa contiene los componentes necesarios para cifrar y realizar las operaciones de seguridad.

Nombre	Encode
Tipo	Clase extática
Padre	-
Descripción	Implementa funciones para encriptar, desencriptar y crear hash.
Métodos	
Nombre	**Descripción**
string GetHash(string str, HashFormat hashFormat)	Dado un parámetro de entrada str de tipo string y el tipo de algoritmo hash a utilizar genera un valor hash.
string GetHexHash(string str,	Dado un parámetro de entrada str de tipo string y el tipo

HashFormat hashFormat)	de algoritmo hash a utilizar genera un valor hash en base 16.
string GetHexHash(byte[] content, HashFormat hashFormat)	Dado un parámetro de entrada content de tipo array de bytes y el tipo de algoritmo hash a utilizar genera un valor hash en base 16.
byte[] GetBinaryHash(string str, HashFormat hashFormat)	Dado un parámetro de entrada str de tipo string y el tipo de algoritmo hash a utilizar genera un valor hash en array de bytes.
byte[] GetBinaryHash(byte[] content, HashFormat hashFormat)	Dado un parámetro de entrada content de tipo array de bytes y el tipo de algoritmo hash a utilizar genera un valor hash en array de bytes.
string EncodeMessage(string message)	Dado un parámetro de entrada se encripta con una contraseña interna.
string EncodeMessage(string message, string password)	Dado un parámetro de entrada se encripta con una contraseña indicada.
byte[] EncodeMessageToBytes(string message)	Dado un parámetro de entrada se encripta con una contraseña interna devolviendo una array de bytes.
byte[] EncodeMessageToBytes(string message, string password)	Dado un parámetro de entrada se encripta con una contraseña indicada devolviendo una array de bytes.
string DecodeMessage(string encryptedMessage)	Dado un parámetro de entrada se desencripta con una contraseña interna.
string DecodeMessage(string encryptedMessage, string password)	Dado un parámetro de entrada se desencripta con una contraseña indicada.
string CreateRandomPassword(int Lenght)	Crea una contraseña alfanumérica con la dimensión indicada.
string EncodeWithAES(string input, string password)	Dado un parámetro de entrada se encripta con el algoritmo AES con una contraseña.
DecodeWithAES(string input, string password)	Dado un parámetro de entrada se desencripta con el algoritmo AES con una contraseña.
string EncodeCustomBase32(int num)	Dado un número realiza una transformación en base 32 personalizada.
int DecodeCustomBase32(string cadena)	Dada una transformación en base 32 personalizada devuelve el número original.

Tabla 60: Clase estática Encode.

Nombre	TokenEncodeInspector
Tipo	Clase
Padre	DelegatingHandler (7)
Descripción	Comprueba la autenticación de una petición http en función de un token de validación.

Métodos	
Nombre	**Descripción**
Task<HttpResponseMessage> SendAsync(HttpRequestMessage request, CancellationToken cancellationToken)	Sobre escritura del padre. Comprueba la autenticación del token a partir de datos del usuario.

Tabla 61: Clase TokenEncodeInspector.

Nombre	TokenCertificateInspector
Tipo	Clase
Padre	DelegatingHandler (7)
Descripción	Comprueba la autenticación de una petición http en función de un token de validación.
Métodos	
Nombre	**Descripción**
Task<HttpResponseMessage> SendAsync(HttpRequestMessage request, CancellationToken cancellationToken)	Sobre escritura del padre. Comprueba la autenticación a partir de un certificado y los datos del usuario.

Tabla 62: Clase TokenCertificateInspector.

Nombre	HttpsGuard
Tipo	Clase
Padre	DelegatingHandler (7)
Descripción	Comprueba que sea una conexión https
Métodos	
Nombre	**Descripción**
Task<HttpResponseMessage> SendAsync(HttpRequestMessage request, CancellationToken cancellationToken)	Sobre escritura del padre. Comprueba que sea una conexión https.

Tabla 63: Clase HttpsGuard.

Nombre	CryptographyHelper
Tipo	Clase
Padre	-
Descripción	Realiza encriptaciones utilizando un certificado.
Métodos	

Nombre	Descripción
X509Certificate2 GetX509Certificate(string subjectName)	Obtiene el certificado con el nombre indicado.
string Encrypt(X509Certificate2 certificate, string plainToken)	Encripta un token en con el certificado indicado.
string Decrypt(X509Certificate2 certificate, string encryptedToken)	Desencripta un token en con el certificado indicado.

Tabla 64: Clase CryptographyHelper.

11.6.5. Capa de utilidades

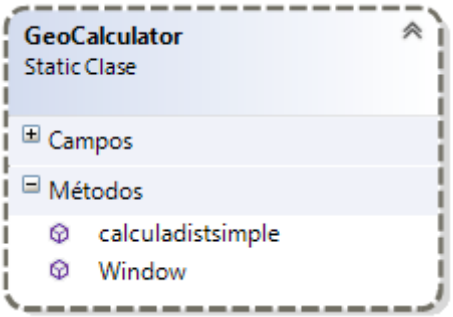


Ilustración 19: Diagrama de capas de las utilidades de la capa infraestructura.

En esta capa se definen aquellos componentes de uso específico que no se pueden clasificar dentro de las otras capas.

Nombre	GeoCalculator
Tipo	Clase estática
Padre	-
Descripción	Realiza cálculos geográficos de baja precisión.
Métodos	
Nombre	**Descripción**
double calculadistsimple(double lat1, double lon1, double lat2, double lon2)	Dadas dos coordenadas geográficas, la primera compuesta por lat1 (latitud) y lon1 (longitud) y la segunda por lat2 (latitud) y lon2 (longitud) calcula la distancia en metros.
Void Window(double lat1, double lon1, double distancia, out double	Dada una coordenada geográfica compuesta por lat1 (latitud) y lon1 (longitud) como punto central y una

latSup, out double lonSup, out double latInf, out double lonInf)	distancia en metros devuelve dos puntos geográficos que componen una ventana rectangular con la distancia indicada.

Tabla 65: Clase GeoCalculator.

11.7. Capa de servicios distribuidos

Esta capa es la que se encarga de proporcionar un servicio web para la comunicación de los dispositivos móviles y clientes del servicio. Para consumir sus recursos se han de realizar peticiones http a los controladores y métodos que se desea utilizando el verbo http (GET, POST, PUT, DELETE) correcto.

11.7.1. Capa de controladores

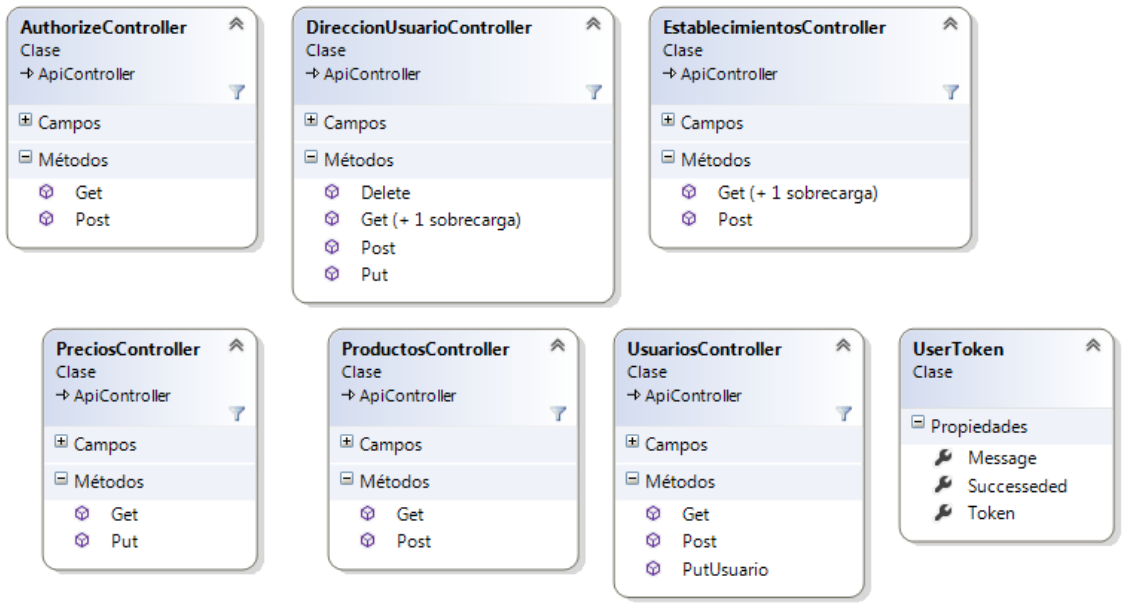


Ilustración 20: Diagrama capa de controladores.

Nombre	AuthorizeController
Tipo	Clase
Padre	ApiController
Descripción	Realiza las tareas de autenticación y obtención del token de autenticación.
Métodos	

Nombre	Descripción
UserToken Get(string nick, string pwd)	Autentifica un usuario y obtiene el token de autenticación por GET.
UserToken Post(UsuarioDTO user)	Autentifica un usuario y obtiene el token de autenticación por POST.

Tabla 66: Clase controladora AuthorizeController.

Nombre	UsuarioController
Tipo	Clase
Padre	ApiController
Descripción	Realiza las operaciones de una entidad usuario.
Métodos	

Nombre	Descripción
UsuarioDTO Get()	Obtiene el usuario autentificado. Es necesario que este autentificado previamente por GET.
void Post(UsuarioDTO usuario)	Añadir un usuario por POST.
public void PutUsuario(UsuarioDTO producto)	Actualizar un usuario. Es necesario que este autentificado previamente y el usuario a actualizar corresponda con el autentificado por PUT.

Tabla 67: Clase controladora UsuarioController.

Nombre	ProductosController
Tipo	Clase
Padre	ApiController
Descripción	Realiza las operaciones de una entidad producto.
Métodos	

Nombre	Descripción
IEnumerable<ProductoDTO> Get(string term)	Devuelve una lista de productos que coincidan total o parcialmente con el término introducido. Es necesario que este autentificado previamente por GET.
void Post(string nombreProducto, string idEstablecimiento, PrecioDTO precioDto)	Añadir un producto relacionándolo con un precio y un establecimiento. Es necesario que este autentificado previamente por POST.

Tabla 68: Clase controladora ProductosController.

Nombre	PreciosController
Tipo	Clase
Padre	ApiController
Descripción	Realiza las operaciones de una entidad precio. Es necesario que este autentificado previamente.
Métodos	
Nombre	**Descripción**
IEnumerable<PrecioDTO> Get(string nombre, double latitud, double longitud, double distancia)	Obtener los precios dentro de una zona geográfica a partir del nombre del producto, una coordenada geográfica (latitud y longitud) y una distancia en metros por GET.
Void Put(string nombre, string idEstablecimiento, PrecioDTO precio)	Actualiza un precio a partir del nombre del producto con el que está relacionado, el identificador del establecimiento con el que está relacionado y el precio por PUT.

Tabla 69: Clase controladora PreciosController.

Nombre	DireccionUsuarioController
Tipo	Clase
Padre	ApiController
Descripción	Realiza las operaciones de las direcciones relacionadas con el usuario. Es necesario que este autentificado previamente.
Métodos	
Nombre	**Descripción**
IEnumerable<DireccionDTO> Get()	Devuelve todo el listado de direcciones del usuario autentificado por GET.
DireccionDTO Get(double latitud, double longitud)	Obtiene la dirección del usuario autentificado a partir de sus coordenadas geográficas por GET.
void Post(DireccionDTO direccionDto)	Añade una nueva dirección al usuario autentificado por POST.
void Put(DireccionDTO direccionDto)	Editar la dirección del usuario autentificado por PUT.
void Delete(double latitud, double longitud)	Borra una dirección del usuario autentificado por DELETE.

Tabla 70: Clase controladora DireccionUsuarioController.

Nombre	EstablecimientoController
Tipo	Clase
Padre	ApiController
Descripción	Realiza las operaciones necesarias para la entidad establecimiento. Es necesario que este autentificado previamente.
Métodos	

Nombre	**Descripción**
IEnumerable<EstablecimientoDTO> Get(double latitud, double longitud, double distancia)	Devuelve todo el listado de establecimientos dentro de una zona geográfica a partir de una coordenada geográfica (latitud y longitud) y una distancia en metros por GET.
EstablecimientoDTO Get(string id)	Obtiene el establecimiento a partir de su identificador por GET.
void Post(EstablecimientoDTO establecimiento)	Añade un nuevo establecimiento por POST.

Tabla 71: Clase controladora EstablecimientoController.

11.8. Capa de presentación

11.8.1. Aplicación web

La función principal de la aplicación web es representar el contenido utilizando las librerías generadas para construir un prototipo de comunidad virtual.

Capa de representación del Modelo

Esta capa es similar a las entidades del dominio, su finalidad principal es poder representar en las vistas el modelo de datos.

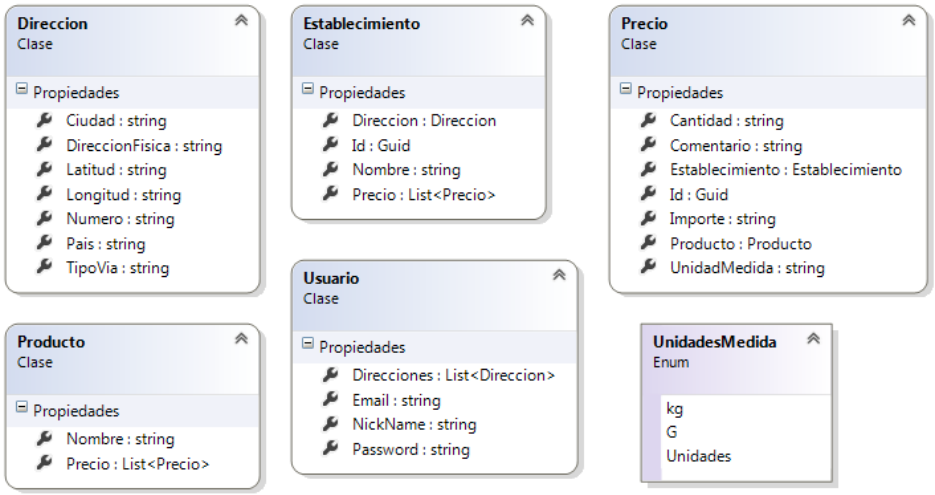


Ilustración 21: Diagrama de representación del modelo de datos en la aplicación web.

Capa de controladores

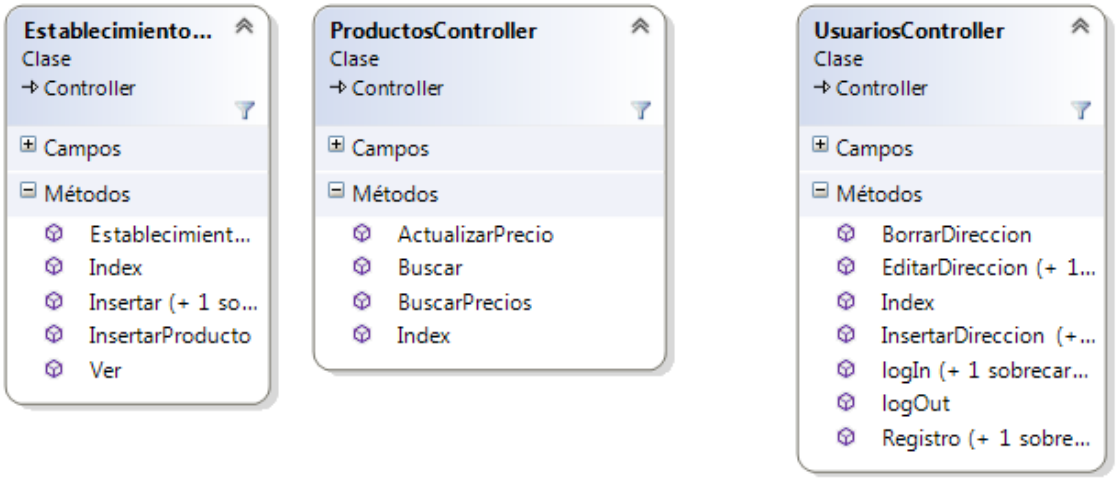


Ilustración 22: Diagrama de controladores de la aplicación web.

Nombre	UsuariosController
Tipo	Clase
Padre	Controller
Descripción	Realiza las operaciones y casos de uso de un usuario.
Métodos	

Nombre	**Descripción**
ActionResult Index()	Obtiene la información del usuario autentificado que luego se representara en la vista. Se ha de llamar por GET. Es necesario que este autentificado previamente.
ActionResult logIn()	Es la acción que mostrara en la vista del formulario para poder autentificarse. Se ha de llamar por GET.
ActionResult logIn(Usuario user)	Es la acción que llamara al formulario de autenticación para validar al usuario y si el usuario es válido lo redirige a la página de búsqueda de productos. Se ha de llamar por POST.
ActionResult Registro()	Es la acción que mostrara en la vista del formulario para poder registrarse (añadir un usuario). Se ha de llamar por GET.
ActionResult Registro(Usuario user)	Es la acción que llamara al formulario de registro para validar al usuario, si el usuario es correcto llama a la acción o método logIn. Se ha de llamar por POST.
ActionResult logOut()	Es la acción que desconectara al usuario y lo redirigirá a la página de logIn. Se ha de llamar por GET.
ActionResult InsertarDireccion()	Es la acción que mostrara en la vista del formulario para poder añadir una dirección al usuario. Se ha de llamar por GET.
ActionResult InsertarDireccion(Usuario usuario)	Es la acción que llamara el formulario de añadir dirección al usuario. Se ha de llamar por POST.
ActionResult EditarDireccion(double latitud, double longitud)	Es la acción que mostrara la vista del formulario para poder editar una dirección al usuario. Se ha de llamar por GET.
ActionResult EditarDireccion(Usuario usuario)	Es la acción que llamara el formulario de editar una dirección del usuario. Se ha de llamar por POST.
JsonResult BorrarDireccion(string	Es la acción que borrara (desasociara) una dirección del

latitud, string longitud)	usuario. Se ha de llamar por POST.

Tabla 72: Clase controladora de la web UsuariosController.

Nombre	ProductosController
Tipo	Clase
Padre	Controller
Descripción	Realiza las operaciones y casos de uso necesarios del producto. Es necesario que este autentificado previamente.
Métodos	
Nombre	**Descripción**
ActionResult Index()	Obtiene la información necesaria para la vista de búsqueda de productos. Se ha de llamar por GET.
JsonResult Buscar(string term)	Es la acción que se llamara para buscar un producto. Se ha de llamar por POST.
JsonResult BuscarPrecios(string nombre, string latitud, string longitud, string distancia)	Es la acción que se llamara para buscar los precios de un producto dando un punto geográfico y una distancia en metros. Se ha de llamar por POST.
JsonResult ActualizarPrecio(string nombre, string idEstablecimiento, Precio precio)	Es la acción que se llamara para actualizar el precio de un producto de un establecimiento en concreto. Se ha de llamar por POST.

Tabla 73: Clase controladora de la web ProductosController.

Nombre	EstablecimientosController
Tipo	Clase
Padre	Controller
Descripción	Realiza las operaciones y casos de uso necesarios del establecimiento. Es necesario que este autentificado previamente.
Métodos	
Nombre	**Descripción**
ActionResult Index(string nombre)	Obtiene la información necesaria para la vista de búsqueda de establecimientos, si se le indica un nombre del producto luego se podrán realizar operaciones relacionas con ese producto en concreto. Se ha de llamar por GET.

ActionResult Ver(string id)	Obtiene la información necesaria para la vista de que mostrara un establecimiento en concreto al indicar un identificador de establecimiento valido por parámetro. Se ha de llamar por GET.
JsonResult BuscarEstablecimientos(string latitud, string longitud, string distancia)	Es la acción que se llamara para obtener los establecimientos introduciendo como parámetros la coordenada geográfica y una distancia en metros. Se ha de llamar por POST.
ActionResult Insertar()	Es la acción que mostrara la vista del formulario para poder añadir un establecimiento. Se ha de llamar por GET.
ActionResult InsertarProducto(string nombreProducto, string idEstablecimiento, Precio precio)	Es la acción que llamara el formulario de añadir un establecimiento. Se ha de llamar por POST.

Tabla 74: Clase controladora de la web EstablecimientosController.

Capa de vistas

Vistas del controlador UsuariosController

Vista LogIn

Ilustración 23: Vista LogIn aplicación web.

Vista Registro

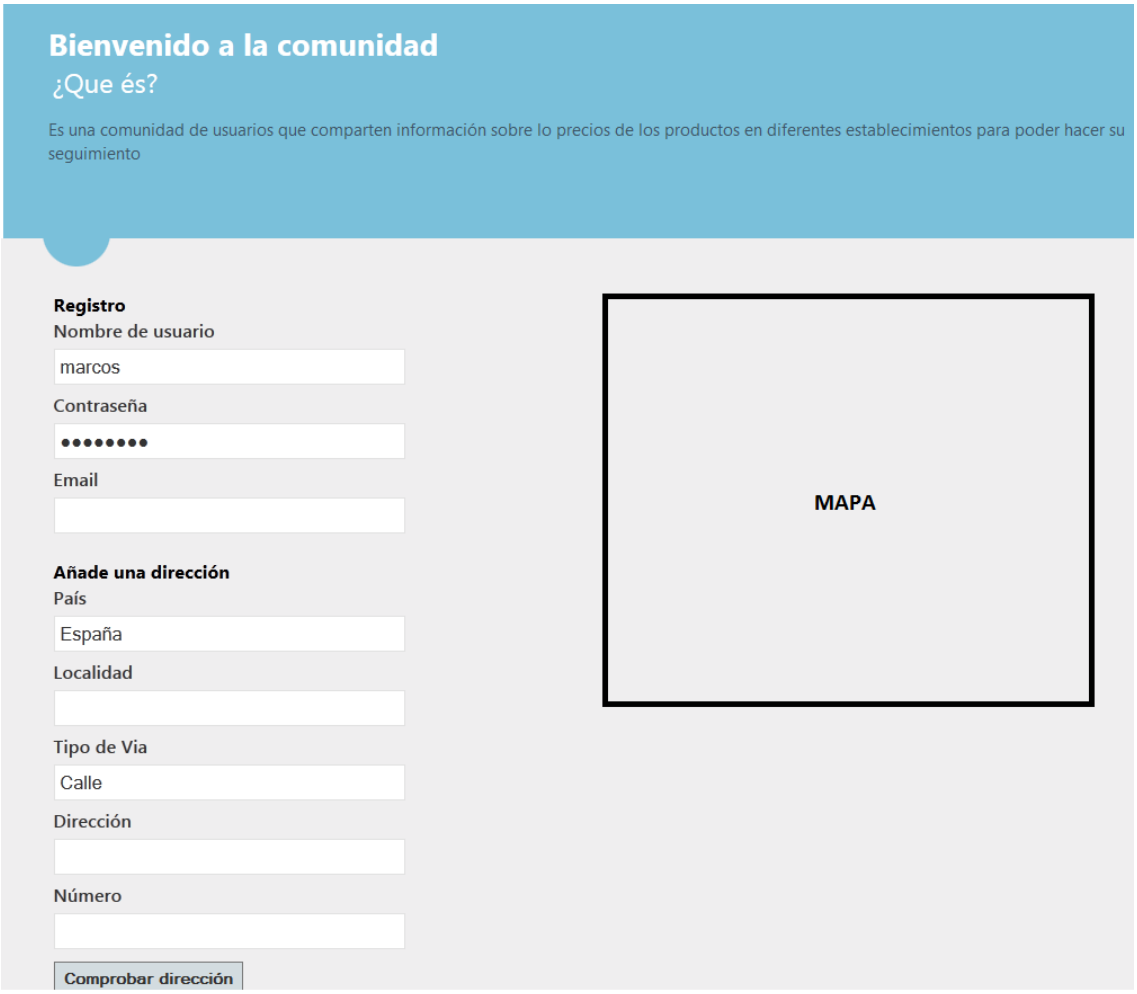


Ilustración 24: Vista Registro de UsuariosController de la aplicación web.

Vista Index

Ilustración 25: Vista Index de UsuariosController de la aplicación web.

Vista EditarDireccion

Ilustración 26: Vista EditarDireccion de UsuariosController de la aplicación web.

Vista InsertarDireccion

Ilustración 27: Vista InsertarDireccion de UsuariosController de la aplicación web.

Vistas del controlador ProductosController

Vista Index

Ilustración 28: Vista Index de ProductosController de la aplicación web.

Vistas del controlador EstablecimientosController

Vista Index

Ilustración 29: Vista Index de EstablecimientosController de la aplicación web.

Vista Ver

Ilustración 30: Vista Ver de EstablecimientosController de la aplicación web mostrando resultados.

Ilustración 31: Vista Ver de EstablecimientosController de la aplicación web para añadir producto o actualizar precio.

Vista insertar

Ilustración 32: Vista Insertar de EstablecimientosController de la aplicación web.

11.8.2. Cliente Android

Debida a la naturaleza propia de Android la aplicación se ha implementado en JAVA.

Capa de representación del Modelo

Esta capa es similar a las entidades del dominio, su finalidad principal es poder representar en las vistas el modelo de datos.

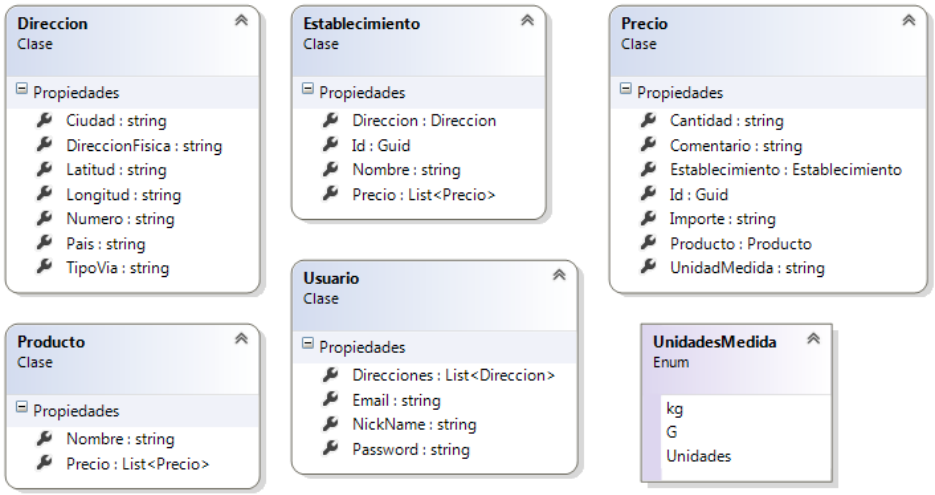


Ilustración 33: Diagrama de la capa de representación del modelo en Android.

Capa de Agentes

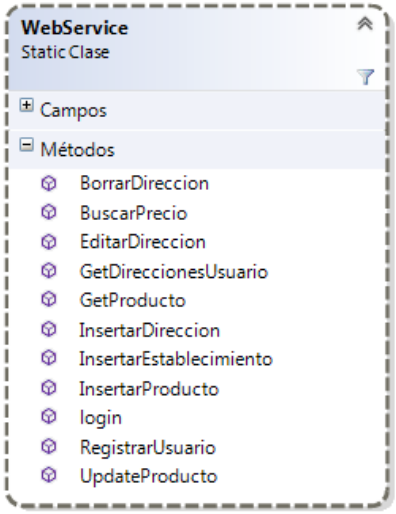


Ilustración 34: Diagrama de agentes del cliente Android.

Nombre	WebService
Tipo	Clase estática
Padre	-
Descripción	Realiza las comunicaciones con el servicio distribuido.
Métodos	

Nombre	**Descripción**
UserToken login(Usuario user)	Realiza la autenticación del usuario.
List<Producto> GetProducto(String token, String term)	Obtienes los productos que coincidan parcial o totalmente con el parámetro de entrada “term”. El parámetro token es el token de autenticación.
List<Direccion> GetDireccionesUsuario(String token)	Obtiene las direcciones del usuario. El parámetro token es el token de autenticación.
void UpdateProducto(String token, String nombre, String idEstablecimiento, Precio precio)	Actualiza el precio de un producto. El parámetro token es el token de autenticación.
void InsertarProducto(String token, String nombre, String idEstablecimiento, Precio precio)	Añade un nuevo producto. El parámetro token es el token de autenticación.
List<Precio> BuscarPrecio(String token, String nombre, double latitud, double longitud, double distancia)	Busca los precios de un producto a partir de una coordenada y una distancia en metros. El parámetro token es el token de autenticación.
void EditarDireccion(String token, Direccion direcion)	Edita una dirección del usuario. El parámetro token es el token de autenticación.
void InsertarDireccion(String token, Direccion direcion)	Añade una dirección del usuario. El parámetro token es el token de autenticación.
void BorrarDireccion(String token, Direccion usuario)	Borra una dirección del usuario. El parámetro token es el token de autenticación.
void InsertarEstablecimiento(String token, Establecimiendo usuario)	Añade un nuevo establecimiento. El parámetro token es el token de autenticación.

Tabla 75: Clase WebService del cliente Android.

Capa de controladores

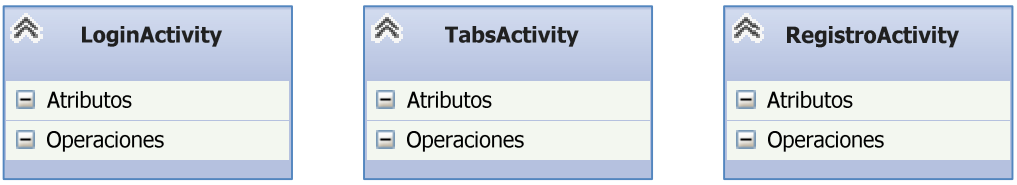


Ilustración 35: Diagrama de controladores del cliente Android.

Debido a que todos sus métodos son privados no se detalla su descripción.

Nombre	LoginActivity
Tipo	Clase
Padre	Activity
Descripción	Gestiona la vista del formulario de autenticación (login).

Tabla 76: Clase LoginActivity del cliente Android.

Nombre	TabsActivity
Tipo	Clase
Padre	Activity
Descripción	Gestiona las diferentes tabs de la aplicación android.

Tabla 77: Clase TabsActivity del cliente Android.

Nombre	RegistroActivity
Tipo	Clase
Padre	Activity
Descripción	Gestiona la vista del formulario de registro de un usuario.

Tabla 78: Clase RegistroActivity del cliente Android.

Capa Vistas

Vista de login

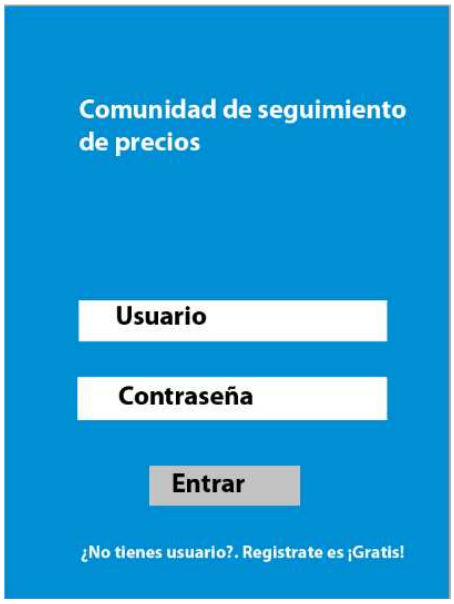


Ilustración 36: Vista login del cliente Android.

Vista Productos

Ilustración 37: Vista tab productos del cliente Android.

Vista Establecimiento

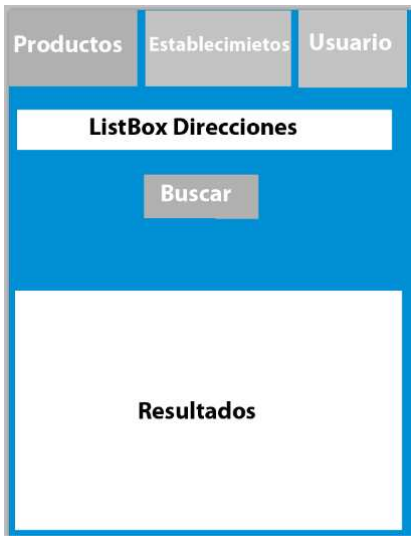


Ilustración 38: Vista tab establecimiento.

Vista Usuario

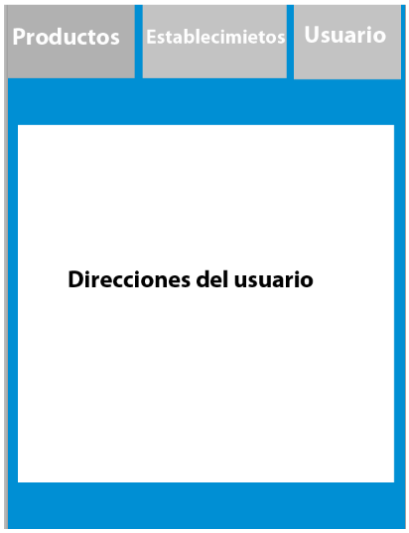


Ilustración 39: Vista tab del usuario del cliente Android.

Vista Registro

Ilustración 40: Vista Registro del cliente Android.

12. Planificación

El proyecto se divide en cuatro grandes fases que se describen a continuación.

- **Planteamiento y diseño**.
  - **Ingeniería de requisitos**: Se realizara la ingeniería de requisitos donde se especificara las necesidades que debe de cumplir el sistema software con sus casos de uso generando la documentación necesaria.
  - **Análisis conceptual**: Una vez terminado el punto anterior se realizara el modelo conceptual para cumplir con los requisitos y casos de uso planteados en el documento de ingeniera de requisitos.
  - **Diseño**: Con el análisis conceptual terminado se realizara el diseño del sistema software aplicando los patrones de diseño necesarios y se generara la documentación necesaria.
  - **WebClient**: Se realizara el diseño necesario para que el cliente web se pueda comunicar correctamente con el servicio distribuido (el servidor).

  - **Cliente Android**: Se realizara el diseño necesario para que el cliente web se pueda comunicar correctamente con el servicio distribuido (el servidor).

- **Implementación de la infraestructura inicial (SaaS).**
  - **Implementación de Contratos**: Desarrollo de los contratos entre las diferentes capas.
  - **Implementación Capa de Dominio**: Desarrollo de la capa de dominio plasmado en la etapa de diseño.
  - **Implementación Capa de datos**: Desarrollo de la capa de datos plasmada en la etapa de diseño.
  - **Implementación Capa de la Aplicación**: Desarrollo de la capa de la aplicación plasmada en la etapa de diseño.
  - **Implementación Capa de servicios distribuidos**: Desarrollo de la capa de la aplicación plasmada en la etapa de diseño.

- **Implementación de la aplicación web (WebClient).**
  - **Implementación de Controladores**: Desarrollo de los controladores y componentes necesarios para que el cliente web pueda funcionar correctamente cogiendo la definición realizada en la etapa de WebClient en el planteamiento y diseño.
  - **Comunicación con SaaS**: Desarrollo de componentes necesarios para la comunicación con el servidor (SaaS), definidos en la etapa de WebClient en el planteamiento y diseño.
  - **Pantallas**: Implementación de la interfaz de usuario.

- **Implementación de la aplicación móvil (Cliente Android).**
  - **Implementación de Controladores**: Desarrollo de los controladores y componentes necesarios para que el cliente android pueda funcionar correctamente cogiendo la definición realizada en la etapa de Cliente Android en el planteamiento y diseño.
  - **Comunicación con SaaS**: Desarrollo de componentes necesarios para la comunicación con el servidor (SaaS), definidos en la etapa de Android en el planteamiento y diseño.
  - **Pantallas**: Implementación de la interfaz de usuario.

12.1. Planificación inicial

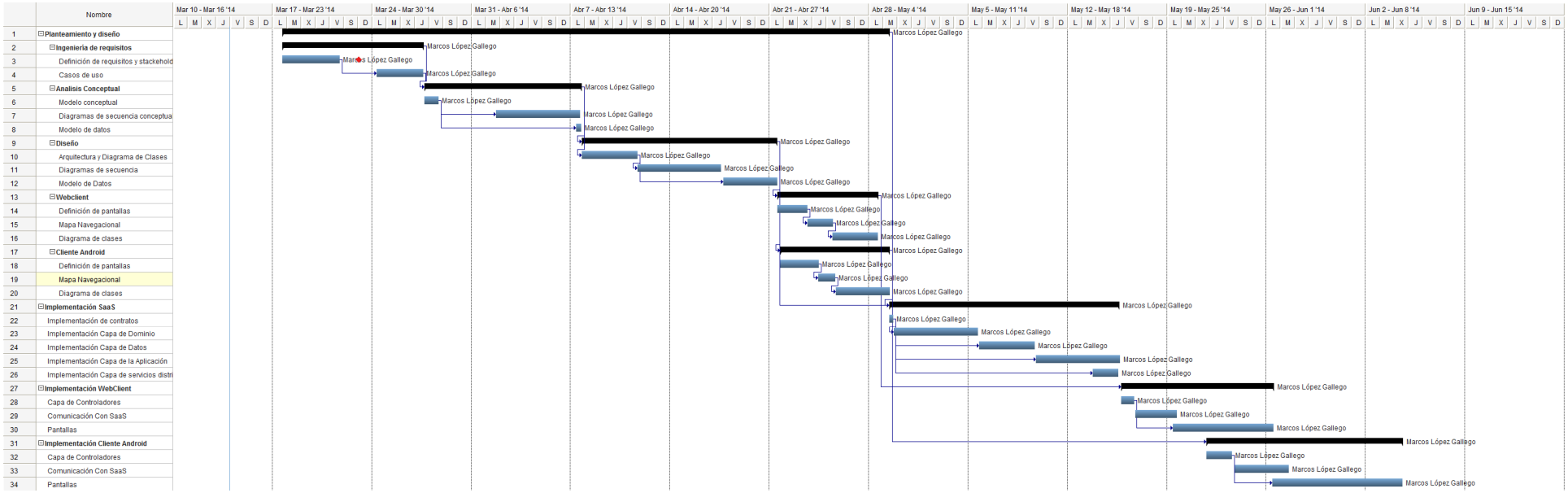
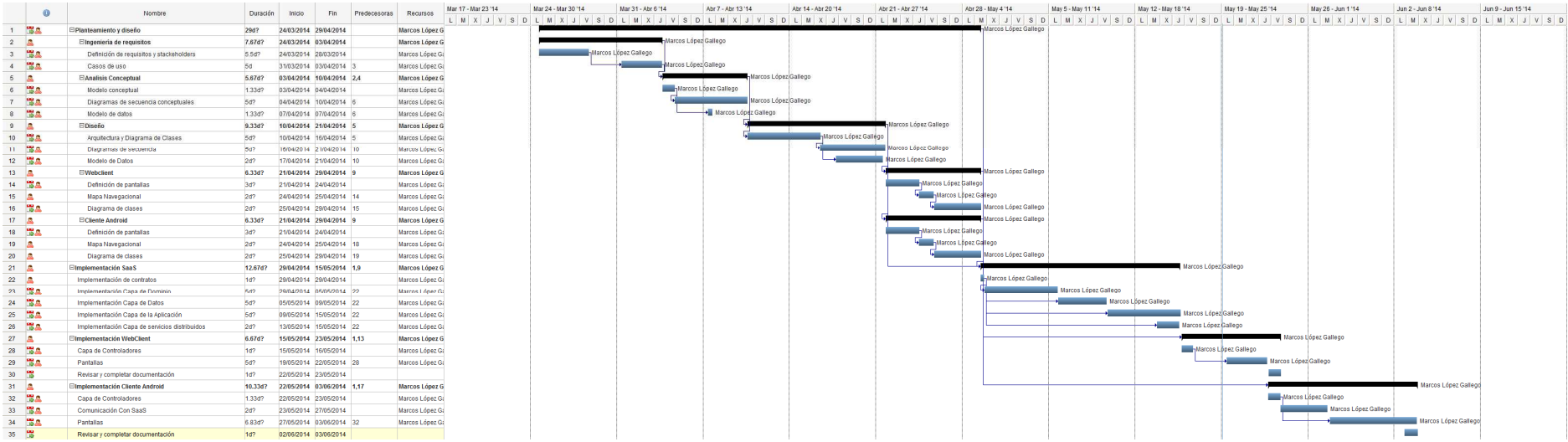


Ilustración 41: Planificación Inicial.

12.2. Planificación revisada



	Nombre	Duración	Inicio	Fin	Predecesoras	Recursos
1	**Planteamiento y diseño**	**29d?**	**24/03/2014**	**29/04/2014**		**Marcos López G**
2	**Ingeniería de requisitos**	**7.67d?**	**24/03/2014**	**03/04/2014**		**Marcos López G**
3	Definición de requisitos y stackeholders	5.5d?	24/03/2014	28/03/2014		Marcos López G
4	Casos de uso	5d	31/03/2014	03/04/2014	3	Marcos López G
5	**Analisis Conceptual**	**5.67d?**	**03/04/2014**	**10/04/2014**	**2,4**	**Marcos López G**
6	Modelo conceptual	1.33d?	03/04/2014	04/04/2014		Marcos López G
7	Diagramas de secuencia conceptuales	5d?	04/04/2014	10/04/2014	6	Marcos López G
8	Modelo de datos	1.33d?	07/04/2014	07/04/2014	6	Marcos López G
9	**Diseño**	**9.33d?**	**10/04/2014**	**21/04/2014**	**5**	**Marcos López G**
10	Arquitectura y Diagrama de Clases	5d?	10/04/2014	16/04/2014	5	Marcos López G
11	Diagramas de secuencia	5d?	16/04/2014	21/04/2014	10	Marcos López G
12	Modelo de Datos	2d?	17/04/2014	21/04/2014	10	Marcos López G
13	**Webclient**	**6.33d?**	**21/04/2014**	**29/04/2014**	**9**	**Marcos López G**
14	Definición de pantallas	3d?	21/04/2014	24/04/2014		Marcos López G
15	Mapa Navegacional	2d?	24/04/2014	25/04/2014	14	Marcos López G
16	Diagrama de clases	2d?	25/04/2014	29/04/2014	15	Marcos López G
17	**Cliente Android**	**6.33d?**	**21/04/2014**	**29/04/2014**	**9**	**Marcos López G**
18	Definición de pantallas	3d?	21/04/2014	24/04/2014		Marcos López G
19	Mapa Navegacional	2d?	24/04/2014	25/04/2014	18	Marcos López G
20	Diagrama de clases	2d?	25/04/2014	29/04/2014	19	Marcos López G
21	**Implementación SaaS**	**12.67d?**	**29/04/2014**	**15/05/2014**	**1,9**	**Marcos López G**
22	Implementación de contratos	1d?	29/04/2014	29/04/2014		Marcos López G
23	Implementación Capa de Dominio	5d?	29/04/2014	06/05/2014	22	Marcos López G
24	Implementación Capa de Datos	5d?	05/05/2014	09/05/2014	22	Marcos López G
25	Implementación Capa de la Aplicación	5d?	09/05/2014	15/05/2014	22	Marcos López G
26	Implementación Capa de servicios distribuidos	2d?	13/05/2014	15/05/2014	22	Marcos López G
27	**Implementación WebClient**	**6.67d?**	**15/05/2014**	**23/05/2014**	**1,13**	**Marcos López G**
28	Capa de Controladores	1d?	15/05/2014	16/05/2014		Marcos López G
29	Pantallas	5d?	19/05/2014	22/05/2014	28	Marcos López G
30	Revisar y completar documentación	1d?	22/05/2014	23/05/2014		
31	**Implementación Cliente Android**	**10.33d?**	**22/05/2014**	**03/06/2014**	**1,17**	**Marcos López G**
32	Capa de Controladores	1.33d?	22/05/2014	23/05/2014		Marcos López G
33	Comunicación Con SaaS	2d?	23/05/2014	27/05/2014		Marcos López G
34	Pantallas	6.83d?	27/05/2014	03/06/2014	32	Marcos López G
35	Revisar y completar documentación	1d?	02/06/2014	03/06/2014		

Ilustración 42: Planificación revisada.

Modificaciones	Justificación
Fase de implementación del webclient. Se ha eliminado la implementación de la comunicación SaaS.	Una vez definida la arquitectura se ha optado por la utilización de las librerías generadas directamente. Se consigue las siguientes mejoras. • Disminución del tiempo de respuesta del webclient. • Liberar el volumen de tráfico y conexiones al sistema SaaS.
Al final de las fases de implementación del webclient y el cliente Android se ha añadido una tarea de revisión del documento de la memoria.	Modificar los posibles cambios originados de la documentación inicial generando su respectivo diagrama de clases por capas.

La planificación ha sido modificada pero al eliminar una tarea planificada y añadida otra, la fecha de la finalización del proyecto se mantiene teniendo margen de desviación por posibles obstáculos.

12.3. Valoración de alternativas y plan de acción

Las tareas en las que se compone el proyecto están bastante ligadas, con lo que un posible retraso implicaría tener que acortar los días, ya sea aumentando las horas diarias o acortar la tarea, en la realización de la tarea siguiente para poder finalizarlo en el plazo previsto.

En la etapa de planteamiento y diseño se ha realizado poniendo más días de lo que inicialmente se podría planificar, para poder alargar aquellas tareas dentro de esta fase, en las que la previsión no haya sido correcta.

En la planificación general se deja las últimas semanas de junio como margen para aquellas tareas que se dilaten en exceso en el tiempo.

En el caso de que la desviación de alguna de las tareas sea superior a 5 días se realizara un plan de acción en función de los siguientes casos.

1. **Fase de planteamiento y diseño**: Priorizar el trabajo imprescindible y suprimir el resto.
2. **Implementación de la infraestructura inicial:** Reducir al máximo el numero de test a realizar. Se realizaran aquellos que sean imprescindible para comprobar el caso base.
3. **Implementación de la aplicación web e Implementación de la aplicación móvil:** Reducir al máximo el numero de test a realizar. Se realizaran aquellos que sean imprescindible para comprobar los casos base. También se reducirá el número de horas en realizar una interfaz gráfica atractiva.

13. Presupuesto

13.1. Consideraciones

En este apartado, se describe una estimación del coste de este proyecto teniendo en cuenta las amortizaciones correspondientes a los recursos de hardware y software. Todos los costes descritos se incluyen los impuestos asociados.

13.2. Control presupuestario

Con el fin de limitar los recursos consumidos, al final de cada sprint el presupuesto se actualizará con el importe efectivo total de horas. Si al finalizar cada sprint se aprecia una desviación presupuestaria superior al equivalente a 1 semana, se establecerá el plan de acción correspondiente descrito en el apartado "Valoración de alternativas y plan de acción".

13.3. Presupuesto de recursos humanos

Este proyecto va a ser desarrollado por una sola persona. Por lo tanto, esta persona tendrá que ser a la vez un gestor de proyecto, ingeniero del software y desarrollador.

Personas	Estimación de horas	Estimación de coste por hora	Total estimado
Marcos López	375	30 €/hora	11.250 €
Total	**375**		**11.250 €**

Tabla 79: Presupuesto de recursos humanos.

13.4. Presupuesto del hardware

Con el fin de ser capaces de diseñar, implementar y probar todas las funcionalidades de las aplicaciones, será necesario un conjunto de hardware para diferentes propósitos. En la Tabla, se proporciona una estimación del coste del hardware necesario teniendo en cuenta su vida útil, así como sus amortizaciones.

Producto	Coste	Unidad	Vida útil	Amortización total estimada
Portátil Dell	700 €	1	5 años	140 €
Total				**140 €**

Tabla 80: Presupuesto del hardware.

13.5. Presupuesto del software

Para realizar el proyecto se necesitarán algunos productos de software para poder llevarlo a cabo. Aunque algunos de ellos están disponibles de forma gratuita, por lo tanto no se contemplan en el presupuesto. Al igual que en el presupuesto de hardware, sus amortizaciones han sido tenidas en cuenta.

Producto	Coste	Unidad	Vida útil	Amortización total estimada
Visual Studio 2013 Profesional	387 €	1	5	77,4 €
Android Studio	0	1	5	0 €
Windows 7	Incluido en Portátil Dell	0		0 €
Total				**77,4 €**

Tabla 81: Presupuesto del software.

13.6. Presupuesto licencias

Producto	Coste Estimado	Unidades	Vida útil	Total estimado
Desarrollador Android	25 €	1	Indefinido	25 €
Total				**25 €**

Tabla 82: Presupuesto licencias.

13.7. Presupuesto de los proveedores

El proyecto, debido a que es una aplicación que debe estar accesible por internet se necesita la contratación de proveedores.

Servicio	Proveedor	Coste mensual	Coste anual
Servidor + dominio	1and1	12,08 €	145 €
Total		12,08 €	**145 €**

Tabla 83: Presupuesto de los proveedores.

13.8. Costes imprevistos

Como costes imprevisto se considerara un posible fallo en el equipo informático durante la realización del proyecto, para él cual se estimara un coste de reparación de aproximadamente 200 € y un posible retraso de 5 días, aproximadamente 185 € de coste, haciendo un total de 380 €.

13.9. Presupuesto total y viabilidad

Concepto	Coste Estimado
Recursos Humanos	11.250 €
Hardware	140 €
Proveedores	145 €
Software	77,4 €
Licencias	25€
Costes imprevistos	380 €
Total	**12.017,4 €**

La estimación inicial es de 12.017,4 €. Al ser un proyecto de un software completo el coste no se puede considerar excesivamente elevado. Como podemos observar el mayor coste es el personal que al realizarse en un entorno académico se reduce a 0 € teniendo un coste más real y estimado de 767,4€. Debido a que no es un coste demasiado elevado, el proyecto se podría rentabilizar fácilmente con publicidad si tiene la aceptación suficiente.

13.10. Impacto social y medioambiental

13.10.1. Impacto social

El proyecto tiene un factor social muy importante. Este proyecto tiene como misión ayudar en la reducción de costes, ayudando a tener un mejor control de la variación de precios entre establecimientos mejorando la competitividad entre establecimientos comerciales y facilitar que las personas con menos recursos económicos puedan llegar a final de mes en un contexto de crisis económica.

13.10.2. Impacto medioambiental

El mayor impacto medioambiental que puede generar el proyecto son los residuos contaminantes generados por la producción de la energía electricidad y fabricación del hardware que necesita el proveedor y el propio equipo. Este tipo de contaminantes no pueden ser directamente controlados sino que las empresas productoras deben de gestionar esos residuos de forma adecuada.

Para intentar disminuir el impacto se ha elegido como proveedor 1and1 al utilizar energías renovables para abastecer sus centros de servidores.

14. Conclusiones y trabajo futuro

14.1. Conclusiones

En el transcurso del proyecto se ha creado toda una arquitectura básica para poder crear una comunidad virtual de seguimiento de precios, un prototipo de aplicación web y un prototipo de aplicación Android.

En el caso de la arquitectura básica se ha podido completar con éxito incluida su capa de distribución de servicios. Para la aplicación web se han podido implementar todos los casos de uso. Pero para la aplicación Android no ha sido posible implementar todos los casos de uso, los casos de uso que no se han podido implementar son todas las que conlleva añadir, editar o borrar una dirección del usuario al haber previsto una curva de reaprendizaje de la implementación de una aplicación Android demasiada optimista.

La principal dificultad que me he encontrado, como ya he comentado, es en el desarrollo de la aplicación Android. Debido a que han pasado aproximadamente 2 años desde la última vez que implemente una. Por esta razón, he tenido que reaprender muchas cosas olvidadas y aprender de nuevas convirtiéndose en un freno considerable para mí en esta fase, obligándome a tomar la decisión de no implementar todos los casos de uso y solo los más relevantes.

Gracias al desarrollo en C# y a la utilización del IDE de Visual Studio, en la mayor parte del proyecto, los tiempos se han podido cumplir, en aquellas donde se han utilizado, debido a que es un lenguaje que dómino a la perfección y utilizo en mi vida profesional. Al haber escogido como ORM el Entity Framework me ha permitido un desarrollo rápido en la capa de acceso a datos (DAL) reduciendo considerablemente el tiempo.

Por último, quería decir que debido a que es un sistema básico se pueden expandir sus funcionalidades con relativamente poco tiempo y esfuerzo. A demás, gracias a la utilización de patrones, sobretodo en el patrón de inyección de dependencias (DI) e inversión del control (IoC), es muy sencillo poder añadirle nuevos componentes.

14.2. Trabajo futuro

Tal como ya he comentado en el apartado anterior, se han realizando dos prototipos. Como trabajo futuro se podría desarrollar una aplicación web y otra Android completas teniendo en cuenta las tendencias actuales del diseño de interfaces y realizando los test oportunos tanto como test de funcionalidad como test de tipo A/B para poder conseguir un sistema lo más estable posible y competitivo.

También se debería de definir un modelo, como por ejemplo, si se quiere convertir en una red social o en un servicio como tal. Para tal cometido, una vez definido el modelo que se desea, habría que revisar los requisitos de calidad (no funcionales) y funcionales para adaptarlos y ampliarlos al modelo elegido.

15. Bibliografía

1. INE. *Nivel, Calidad y condiciones de vida.* [En línea] [Citado el: 4 de 03 de 2014.] http://www.ine.es/prensa/np811.pdf.

2. El comparador. [En línea] Caprabo. http://www.miclubcaprabo.com/es/mi-ahorro/micomparador.

3. El Economista. [En línea] http://www.eleconomista.es/catalunya/noticias/5152363/09/13/Caprabo-compara-sus-precios-con-Mercadona-y-devolvera-la-diferencia-.html#.Kku8fisPAr1V6b2.

4. Carritus. [En línea] http://www.carritus.com/.

5. ¿Qué es Scrum? *Proyectos agiles.* [En línea] [Citado el: 19 de 02 de 2014.] http://www.proyectosagiles.org/que-es-scrum.

6. Team Foundation Server. *visualstudio.* [En línea] [Citado el: 20 de 02 de 2014.] http://www.visualstudio.com/en-us/products/tfs-overview-vs.aspx.

7. **César de la torre, Unai Zorrilla Castro, Javier Calvarro Nelson, Miguel Ángel Ramos Barroso.** *Guía de Arquitectura N-Capas orientada al Dominio con .NET 4.0.* s.l. : Krasis, 2010.

8. MSDN . *DelegatingHandler.* [En línea] http://msdn.microsoft.com/es-es/library/system.net.http.delegatinghandler%28v=vs.118%29.aspx.